МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Уманський державний педагогічний університет імені Павла Тичини

О.І. Жмурко, Т.О. Охріменко

Олімпіади з програмування
Прості задачі

Умань
Візаві
2020

УДК 378.016:004.42
Ж 77

Рецензенти:

Месюра В.І., кандидат технічних наук, професор кафедри комп'ютерних наук Вінницького національного технічного університету, директор українського сайту Південно-східного європейського регіонального півфіналу першості світу з програмування ACM-ICPC, засновник і координатор Всеукраїнської студентської олімпіади з програмування;

Вакалюк Т.А., доктор педагогічних наук, доцент, професор кафедри інженерії програмного забезпечення Державного університету «Житомирська політехніка».

Рекомендовано до друку Вченою радою факультету математики та інформатики
Уманського державного педагогічного університету імені Павла Тичини
(протокол № 3 від 23 квітня 2020 р.)

Жмурко О. І.

Ж 77 Олімпіади з програмування. Прості задачі / О.І. Жмурко, Т.О. Охріменко ; МОН України, Уманський держ. пед. ун-т імені Павла Тичини. – Умань : Візаві, 2020. — 298 с.

Розглянуто програмування простих задач спортивного програмування. Наведені та пояснені розв'язки задач.

Видання може бути корисним студентам та учням при вивченні програмування та підготовці до олімпіад з програмування та інформатики.

УДК 378.016:004.42

Зміст

Вступ

Олімпіадне програмування — це захоплююча частина програмування на правильність, ефективність та швидкість. Завдяки такому це є також різновид спорту, тому часто кажуть про нього як спортивне програмування.

Спортивне програмування — особливий вид інтелектуальних змагань, в яких учасники вирішують одну або декілька алгоритмічних задач за обмежений час за допомогою комп'ютера.

Спортивне програмування — це змагання з програмування на швидкість, коректність, ефективність, це справжня битва людських інтелектів засобами комп'ютерної техніки.

Навчальний посібник призначено, перш за все, для студентів фізико-математичних напрямків та школярів.

Початковий етап — прості задачі повинні розв'язуватись швидко, за рахунок тренувань, без втрат часу, та якісно, без втрат балів.

Посібник можна розглядати як деякий практикум з кодування та алгоритмів.

Метод роботи з книгою має бути традиційним — розв'язуємо задачу, здаємо її перевіряючій системі, тоді знайомимось з наведеним тут кодом програми.

Аналіз чужого коду, його позитивних та негативних сторін, є ефективним методом навчання.

Розділ 1

Олімпіади та ресурси підготовки

Мета змагань з програмування це написання коду комп'ютерних програм, які здатні вирішувати запропоновані завдання. Переважна більшість проблем, що містяться у змаганнях з програмування, є математичними або логічними. Зазвичай ці завдання належать до наступних категорій: комбінаторика, теорія чисел, теорія графів, геометрія, аналіз рядків та структури даних. Проблеми пов'язані зі штучним інтелектом також популярні на деяких змаганнях.

Незалежно від категорії завдання, процес вирішування завдання може бути поділений на два основних етапи: розробка алгоритму та кодування алгоритму обраною мовою програмування (список дозволених мов програмування залежить від конкретного змагання).

На більшості змагань визначення результатів проводиться автоматично за допомогою спеціальних систем. Кожен розв'язок завдання запускається на сервері. На вхід цьому розв'язку подається список тестових прикладів (зазвичай секретний). У більшості випадків вирішення проблем маркуються за принципом «все або нічого», тобто якщо вирішення спрацювало неправильно на хоча б одному з тестових прикладів, воно не зараховується. Однак, деякі змагання використовують процентну систему оцінювання, тобто за розв'язок дають стільки відсотків, скільки відсотків тестових прикладів було розв'язано правильно.

1.1. Олімпіади, підготовка

ICPC

Міжнародна студентська олімпіада з програмування (International Collegiate Programming Contest) [1] [2] або Студентський командний чемпіонат світу з програмування — найбільша студентська командна олімпіада з програмування.

У світі у ICPC приймає участь більше 50 000 студентів, з більше ніж 3 000 університетів, з 111 країн, на більше ніж 400 сайтах змагань.

Змагання проходить між командами з трьох студентів. До участі допускаються студенти вищих навчальних закладів, а також аспіранти першого року навчання. Студенти які двічі брали участь у фінальній стадії олімпіади або п'ятикратно брали участь в регіональному відборі не допускаються до участі. Також є обмеження за віком: учасники старше 24 років не допускаються. В команді допускається один запасний учасник. У команди є тренер (coach).

Тур олімпіади відбувається таким чином: кожній команді видається один комп'ютер і від восьми до дванадцяти завдань різної складності, умови яких написані англійською мовою (на початкових етапах умови можуть бути національними мовою, часто українською та англійською мовами). Змагання триває п'ять годин. Команди створюють програму і відправляють на тестування. Якщо програма видала неправильну відповідь або не вклалася в обмеження за часом або пам'яті, то команда отримує про це повідомлення. Завдання вважається розв'язаним, якщо програма видала правильні відповіді на всіх тестах. На відміну від інших олімпіад, часткові рішення не враховуються. Користування допоміжними джерелами інформації, словниками та електронними пристроями не дозволяється.

Перемагає команда, яка розв'язала найбільше число завдань. Якщо кілька команд рішають однакову кількість задач, то їх місця визначаються штрафним часом.

З 1977 по 2017 рік олімпіада проводилась під егідою організації ACM (Association for Computing Machinery) і була відома під назвою ACM ICPC.

Студентський командний чемпіонат світу з програмування ICPC сягає своїм корінням в змагання, що проводилося в Техаському університеті в 1970. Свій нинішній вигляд чемпіонат прийняв в 1977 році, коли перший фінал був проведений в рамках щорічної конференції ACM з інформатики, і з тих пір проводиться щорічно.

Починаючи з 1989 року, організацією змагань займається університет Бейлора. У різний час спонсорами змагань ставали такі компанії, як Apple, AT&T і Microsoft, з 1997 по 2017 рік генеральним спонсором була компанія IBM, починаючи з 2018 року глобальним спонсором коштів програмування є компанія JetBrains.

ICPC складається з 4-х етапів. All-Ukrainian Collegiate Programming Contest є складовою Southeastern Europe Regional Contest (SEERC), що в свою чергу входить в Europe Regional Contest.

1-й етап — обласний. Регіональні (2-й український етап) — Ukraine Eastern Contest; Ukraine Western Contest; Ukraine Northern Contest;

Ukraine Southern Contest; Ukraine Central Contest; Ukraine Southwestern Contest. Етап проводиться синхронно по всіх сайтах Європи.

Учні шкіл та коледжів також мають можливість прийняти участь а олімпіаді, шкільному дивізіоні. За необхідності вони можуть продовжити участь у команді на наступному етапі вже як студенти 1-го курсу.

Всеукраїнська студентська олімпіада з програмування є компонентою ICPC. Перші три етапи ICPC вважаються етапами Всеукраїнської студентської олімпіади з програмування.

BAPC

Benelux Algorithm Programming Contest [3](BAPC) — контест з програмування для студентів з Бельгії, Нідерландів та Люксембургу. Він організовується щорічно вищим навчальним закладом. З 1991 по 2004 рік конкурс проводився під назвою NKP (голландський чемпіонат з програмування). Офіційний веб-сайт [4].

З 2006 року BAPC є офіційним попереднім NWERC (North Western European Regional Contest), організований ACM. Це означає, що установи-учасники визначають, які команди вони направлятимуть до NWERC, грунтуючись на результатах BAPC. Кращі команди NWERC запрошуються взяти участь у Всесвітньому фіналі ACM-ICPC.

За місяць до BAPC проводиться попередній раунд, щоб визначити, які команди мають право представляти свою установу в BAPC.

Крім рейтингу студентів, існує рейтинг компаній. Однак конкурс однаковий для всіх учасників.

Олімпіади з інформатики

Деякі олімпіади з інформатики побудовані аналогічно до олімпіад з програмування.

IOI

Міжнародна олімпіада з інформатики [5](IOI) — це щорічне змагання з інформатики серед школярів. IOI — одне з найстаріших змагань для школярів, вперше була проведене в 1989 році.

Змагання складається з двох днів, в якому учасникам пропонується вирішити і запрограмувати алгоритмічні завдання. Учасники змагаються індивідуально; від кожної країни може бути не більше чотирьох учасників.

У кожен день змагання учасникам зазвичай пропонується вирішити три-чотири завдання за п'ять годин. Кожен учасник вирішує завдання самостійно, користуючись одним комп'ютером. Суворо забороняється спілкування з іншими учасниками, використання навчальної літератури і т.д.

Зазвичай для вирішення завдання необхідно написати програму на мові C, C++ або Pascal і відіслати її перед закінченням п'ятигодинного змагання. Після змагання програма оцінюється за допомогою запуску її на наборі секретних тестів (зазвичай 10-20 тестів). Учасник отримує бали за кожен правильно вирішений тест, за умови що час роботи програми і обсяг використовуваної нею пам'яті укладаються в зазначені в умові завдання обмеження.

Бали, отримані за два дні змагання, підсумовуються для кожного учасника окремо. На церемонії нагородження учасники нагороджуються відповідно до рейтингу, який будується на основі сумарної кількості балів.

Всеукраїнські олімпіади з інформатики [6].

NetOI центр підтримки та проведення Всеукраїнських олімпіад школярів з інформатики [7].

Інші олімпіади

Також змагання проводять великі компанії:

Google Code Jam — змагання, яке проводиться з 2003 року. Засновано та спонсорується компанією Google.

Facebook Hacker Cup — змагання, яке проводиться з 2011 року. Засновано і спонсорується компанією Facebook.

TopCoder Open — Algorithm — змагання, яке проводиться з 2004 року за підтримки TopCoder. Topcoder Open — неофіційний чемпіонат світу з програмування серед професіоналів.

Архіви задач та тренувальні сервери

Архіви задач з тестуючою системою на сайтах:

acmp [8];

e-olymp [9] (6+ мільйонів розв'язків, 100+ тисяч користувачів, 9+ тисяч задач);

codeforces[10],
timus[11].

Львів, два пов'язаних сайти: Алготестер [12] та ACM Контестер [13].

Деякі тренувальні сайти регулярно проводять олімпіади, участь в яких може бути як лімітована, так і доступна кожному користувачу сайту (див. сайти вказані вище).

1.2. Мови програмування та компілятори

Мови програмування

ICPC Команди пишуть розв'язки мовами програмування C, C++, Java, Python або Kotlin. Під час деяких турів, аж до регіональних, набір мов може бути змінений. Наприклад, додається Pascal, на I етапі олімпіади: C, C++, Java, Pascal, Python, C#.

IOI для вирішення завдання необхідно написати програму на мові C, C ++ або Pascal.

На тренувальних сайтах може бути використана суттєво більша кількість мов програмування.

На сайті e-olymp [9] можна здавати задачі на C/C++,C#, Go, Haskell, Java, JavaScript, Kotlin, Pascal, PHP, Python, Ruby, Rust.

На сайті acmp [8] можна здавати задачі на C/C++, Python3, PyPy3, Pascal, Java, C#, Basic, Go.

На сайті codeforces [10] — C, C++, C++11; Pascal; C#; Java 6, 7, 8; Ruby; Python 2, 3; PHP 5; Haskell; D; OCaml; Scala; JavaScript; Pascal; Perl.

Компілятори

На сайті **acmp** [8] система перевірки використовує наступні компілятори:

MinGW GNU C++ 8.1;
Python 3.7.3; PyPy3 6.0.0;
Free Pascal 3.0.4,
PascalABC.NET 3.5.1;
Java SE JDK 10.0.1
Borland Delphi 7.0
Microsoft Visual C++ 2017
Microsoft Visual C# 2017

Microsoft Visual Basic 2017
Go 1.10.3
На сайті **codeforces.com** [10] GNU C++ 4, GNU C++11 4, GNU C 4, MS VS C++ 2010; MinGW g++.exe (GCC) 4.9.2;
C# .NET, Mono C# compiler version 3.2.3; DMD32 D Compiler v2.064.2;
Delphi 7 [Borland Delphi Version 15.0];
Free Pascal Compiler version 2.6.2;
Haskell Glorious Glasgow, version 7.6.1;
Java 6 javac 1.6.0, Java 7 javac 1.7.0, Java 8 javac 1.8.0
Ocaml ocamlopt 4.00.1
Perl v5.12.2;
PHP 5.3.8;
Python 2.7.8, Python 3.4.1;
Ruby 2.0.0p353;
Scala compiler version 2.11.1;
JavaScript V8 3.23.0;

1.3. Введення-виведення

Є два шляхи введення-виведення.

По-перше використовуються файли. Стандартні назви цих файлів — «**input.txt**» та «**output.txt**».

Також використовуються стандартні потоки введення-виведення — консоль, введення як з клавіатури, виведення як на екран.

Останнім часом перевага надається останньому варіанту.

1.4. Параметри задачі та розв'язку

Кожній задачі виділяється ліміт часу та пам'яті.

Щоб запобігти плутанині використовуємо наступне кодування задач (її номеру) — спочатку літера, що характеризує архів задач, потім чотирицифровий номер задачі в архіві. Для acmp [8] — літера a, наприклад a0001. Для codeforces [10] — літера c, наприклад c0001. e-olymp [9] — літера e, наприклад e0001. Lviv — літера L, наприклад L0001. MIPT — літера m, наприклад m0001. Timus — літера t, наприклад t0001.

Там, де можна було, ми опустили вимоги до вхідних і вихідних даних та їх приклади. Водночас необхідно мати на увазі, що приклади можуть складати невід'ємну частину умови задачі.

Якщо була необхідність, умови було перекладено на українську. Важливо відмітити, що старших та міжнародних етапах умови задач даються англійською.

Складність задач

В системах тренування задачі статистично оцінюються складністю — відсотком користувачів, що не справились з задачею, з поміж користувачів, які відправили код програм (розв'язок) до цієї задачі.

Чотири числа, пов'язані зі складністю це кількість відсилань на перевірку; кількість правильних розв'язків; кількість користувачів, що відправляли розв'язок цієї задачі; Наприклад: складність задачі: 2% — 520/320/263 /259 (e-olymp). Вказані числа не є абсолютними, а відображають стан розв'язання задачі на момент написання цього видання, водночас вони дозволяють орієнтуватись в складності задачі та співвідносити з складністю інших задач.

Розв'язок наближено характеризується часом виконання та об'ємом використаної пам'яті, неточність пов'язана з станом ОС, виконанням інших процесів тощо. Не забуваємо, що з часом можуть поміняти компілятори, їх налагодження, так що числа доволі відносні, але можуть бути в нагоді.

За правилами ACM задача вважається розв'язаною, якщо пройшла всі тести. При тренуваннях на сайтах вказується відсоток тестів, що пройшла задача.

Олімпіади, що проводяться на тренувальних сайтах можуть вибирати і правила, за якими зараховується відсоток пройдених задачею тестів.

При підготовці до олімпіад можна використовувати on-line компілятори, такі як ideone [14], onlinegdb [15].

onlinegdb.com в тому числі C, C++, Python, PHP, Ruby, C#, VB, Perl, JavaScript, Pascal

Характерний ліміт часу — 1 с, ліміт використання пам'яті — 64 MiB (e-olymp). Якщо ліміти були інші, їх відображаємо в умові задачі, стандартні, як правило, опускаємо.

1.5. Готові розв'язки

Є література з підготовки до розв'язання олімпіадних задач: «Практикум програмування Python / C++ на e-olymp.com» [16], «Олимпиадное программирование» [17], «Олимпиадные задачи по программированию» [18], «Решение сложных и олимпиадных задач по программи-

рованию: Учебное пособие» [19], «Спортивное программирование» [20], «Спортивное программирование. Сборник рецептов для программирующих на C++». [21], «Московские олимпиады по информатике» [22], «Олимпиадные задачи по программированию и лучшие решения. Часть 1» [23], «Программирование: теоремы и задачи» [24].

codeforces [10] Зручним може виявитись те, що можна бачити не тільки свої розв'язки але і інших учасників.

ОДУ ім.І.І.Мечнікова Сайт [25] містить розв'язки студентів, в тому числі з e-olymp.com. Коди розв'язків — коди студентів, поновлюються щорічно, водночас будемо наводити деякі з них.

На тренувальному сайті acmp [8] також є навчальна компонента і наводяться розв'язки та коди деяких програм.

В мережі з'являються та зникають сайти з готовими розв'язками олімпіадних задач, частіше це коди програм. Прикладом є [26], [27].

Розділ 2

Відомості, алгоритми, програмування, поради, трюки

Неможливо програмувати без знань з математики, фізики, інформатики тощо, без того що вже винайшли та розробили інші. [1]

2.1. Числові системи

Число є одним з найголовніших об'єктів математики, який використовується для підрахунку, вимірювання та для маркування. Окрім того, що числа використовуються при лічбі та вимірюванні, вони використовуються також для маркування (наприклад, як номер телефону), упорядкування (серійний номер і для кодування книг, товарів, мешканців тощо). Взагалі, термін число може вказувати на символ, слово або математичну абстракцію. З глобальною цифровізацією число може бути пов'язане з будь-яким об'єктом.

Теорія чисел

Теорія чисел або вища арифметика — галузь математики, яка розпочалась з вивчення деяких властивостей натуральних чисел, пов'язаних з питаннями подільності і розв'язання алгебраїчних рівнянь у натуральних (а згодом також цілих) числах [28].

В теорії чисел у широкому розумінні розглядаються як алгебраїчні, так і трансцендентні числа, а також функції різноманітного походження, які пов'язані з арифметикою цілих чисел та їх узагальнень. У дослідженнях з теорії чисел, поряд з елементарними і алгебраїчними методами застосовуються також геометричні і аналітичні.

Більш старий термін для теорії чисел — арифметика . На початку XX століття її витіснила "теорія чисел". (Слово "арифметика" використовується загалом для позначення "елементарних обчислень"; воно також

[1] Більша частина загально-відомої інформації знаходиться в класичних підручниках, а тут скомпільована переважно з Вікіпедії, що опирається на них

набуло інших значень в математичній логіці, як в арифметиці Пеано, і в інформатиці, як в арифметиці з плаваючою комою/крапкою.)

Натуральні числа

Натуральні числа (natural numbers), дослівно — «природні» числа (*лат.* «natura» — природа) — числа, що виникають природним чином при лічбі. Це числа: 1, 2, 3, 4, ... Множину натуральних чисел прийнято позначати знаком $\mathbb{N}$. Натуральні числа — найдавніші числа, які стали використовувати люди. Поняття натурального числа, викликане потребою лічби предметів, виникло ще в доісторичні часи.

Існують два основних підходи до означення натуральних чисел:

- числа, що використовуються при лічбі предметів (перший, другий, третій...) — підхід, загальноприйнятий у більшості країн світу; формалізованим різновидом цього підходу є аксіоматичне описання системи натуральних чисел за допомогою аксіом Пеано;
- числа для позначення кількості предметів (один предмет, два предмети...).

Для позначення множини всіх натуральних чисел використовують символ **N** або $\mathbb{N}$. В старих текстах також іноді використовували символ **J** для позначення цієї множини. Ця множина є нескінченнозліченою: тобто вона є нескінченною і при тому зліченою за визначенням.

Щоб задати однозначно чи включено в цю множину число 0 або ні, іноді в першому випадку додають нижній індекс (або верхній) "0" при більш формальному позначенні, а в другому випадку додають верхній індекс "*" або нижній підпис ">0".

$\mathbb{N}^0 = \mathbb{N}_0 = \{0, 1, 2, \ldots\}$; $\mathbb{N}^* = \mathbb{N}^+ = \mathbb{N}_1 = \mathbb{N}_{>0} = \{1, 2, \ldots\}$.

Альтернативно $\{1, 2, \ldots\} = \mathbb{Z}^+$; $\{0, 1, 2, \ldots\} = \mathbb{Z}^{>0}$.

Беззнакові типи даних в мовах програмування є підмножиною $\mathbb{N}_0$ (обмеженою зверху, скінченною, зліченною).

Від'ємні числа і нуль

Від'ємне число — дійсне число, що менше за нуль. Від'ємні числа розташовані на числовій осі ліворуч від нуля. Протилежне поняття — додатне число.

З'явилось в математиці при розширенні множини натуральних чисел.

Від'ємні числа при запису позначаються спереду знаком мінус: -1,-2, -3,... Для кожного цілого числа a існує і єдине протилежне йому число,

що позначається $-a$, таке що $a + (-a) = 0$. Якщо a додатне, то протилежне йому число — від'ємне, і навпаки. Нуль протилежний самому собі.

0 (нуль, *лат.* nullus — ніякий) — цифра й одночасно число, нульовий елемент математичної структури — нейтральний елемент відносно операції додавання. Множення будь-якого елемента множини на нуль дає нуль.

Від'ємний і додатний нуль — нескінченно малі числа.

Парність Нуль є парним числом оскільки ділиться без залишку на 2. Нуль (за межами радянської школи) — натуральне число, і є єдиним недодатнім натуральним числом.

2.1.1. Цілі числа

Цілі числа — в математиці елементи множини $\mathbb{Z} = \{\ldots -3, -2, -1, 0, 1, 2, 3 \ldots\}$, яка утворюється замиканням натуральних чисел відносно віднімання. Таким чином, цілі числа замкнуті відносно додавання, віднімання та множення.

Дійсне число є цілим, якщо його десяткове подання не містить дробової частини (але може містити знак).

Абсолютною величиною цілого числа a називається це число з відкинутим знаком. Позначення: $|a|$.

Для позначення множини цілих чисел використовується символ $\mathbb{Z}$ (від нім. Zahlen — «числа»), який може в різних авторів використовуватися для позначення групи множин: $\mathbb{Z}^+$, $\mathbb{Z}_+$ або $\mathbb{Z}^>$ для позначення додатних цілих чисел, $\mathbb{Z}^\geq$ для не від'ємних цілих чисел, $\mathbb{Z}^\neq$ для всіх цілих чисел крім нуля. Деякі автори використовують позначення $\mathbb{Z}^*$ для всіх цілих чисел крім нуля, інші для позначення не від'ємних цілих чисел, або для –1, 1. Для множини цілих за модулем p інколи використовують позначення $\mathbb{Z}_p$.

Операції з цілими числами, на відміну від з дійсними, завжди дають точний результат.

При написанні програм всюди, **де можливо** (при виборі ціле – дійсне), **потрібно використовувати цілі числа**. По-перше процесори оперують з ними скоріше, нарешті завжди маємо точний результат.

Різниця між математикою і програмуванням Множина цілих чисел в програмуванні, як правило, є обмеженою. Математичному поняттю цілих відповідає ціле в довгій арифметиці.

Довга арифметика (arbitrary-precision arithmetic, bignum arithmetic, infinite-precision arithmetic, длинная арифметика) означає програмний розрахунок «цифра за цифрою», на відміну від використання процесорних команд, що виконують дії з числом як цілим об'єктом, з розрядами числа паралельним чином.

Довга арифметика є вбудованою компонентою в Python, Ruby та Java. В JavaScript є тип даних BigInt, який дозволяє працювати з цілими числами довільної довжини. В C# для цього є клас BigInteger.

В мовах програмування зі статичною типізацією потрібно акуратно відноситись до вибору типу даних цілого типу. В сучасних процесорах (64 біт) обробка цілих різного типу відбувається практично однаково. Загалом можна взяти за правило використовувати всюди longint, int64, long long, крім параметрів циклів. Тип даних в масивах вибирають з мінімально допустимого, для зменшення розміру масиву в пам'яті.

Типи цілих даних: C/C++ — short, unsigned short, int, unsigned int, long, unsigned long, long long і unsigned long long, деякі компілятори C++ вже підтримують int128; Free Pascal — Byte, ShortInt, Byte, int8, SmallInt, Word, int16, Integer, int32, LongWord, longint, int64; Java — byte, short, char, int, long;

Внутрішнє подання цілих чисел в цифровій техніці — двійковий код, від'ємні — в доповнювальному коді.

Ділення

Подільність (divisibility, делимость) — фундаментальна властивість натуральних та цілих чисел. Число a ділиться на b, відповідно, число b є дільником a, якщо частка $\frac{a}{b}$ — ціле число. Будь-яке натуральне число ділиться на одиницю і на себе. Якщо дане число не має інших дільників, то таке число називається простим, в іншому разі — складеним.

Питання подільності натуральних чисел розглядалися уже в античні часи. Евкліду належить один з найвідоміших результатів математики, твердження, що не існує найбільшого простого числа, тобто множина простих чисел — нескінченна. Він також навів найперший в історії алгоритм, а саме алгоритм Евкліда знаходження найбільшого спільного дільника двох натуральних чисел. Цікаво відзначити, що це — не тільки найдавніший, а й один з найефективніших алгоритмів в математиці, який майже не був вдосконалений за більш ніж дві тисячі років, що минули по тому. Але набагато раніше за Евкліда, Піфагор і піфагорійці

розробили теорію досконалих і дружніх чисел, які відігравали важливу роль у їх філософській системі.

Одиниця має рівно один дільник і не є ні простою, ні складеною. У кожного натурального числа, більшого за одиницю, є хоча б один простий дільник. Власним дільником числа називається всякий його дільник, відмінний від самого числа. У простих чисел існує лише один власний дільник — одиниця.

Незалежно від подільності цілого числа a на ціле число $b \neq 0$, число a завжди можна розділити на b із залишком, тобто представити у вигляді: $a = bq + r$, де $0 \leq r < |b|$. У цьому співвідношенні число q зветься неповною часткою (quotient, частное), а число r — остачею від ділення a на b (remainder, остаток). Як частка, так і остача визначаються однозначно. Число a ділиться без остачі (націло) на b тоді та лише тоді, коли залишок від ділення a на b дорівнює нулю.

Ділення — ділене $\div$ дільник = частка.

Ділення з остачею (mod), ділення за модулем —

ділене mod дільник = остача.

Всяке число, яке ділить як a, так і b, називається їх спільним дільником; максимальне з таких чисел називається найбільшим спільним дільником. У будь-якої пари цілих чисел є принаймні два загальних подільника: +1 та -1. Якщо інших спільних дільників немає, то ці числа називають взаємно простими числами.

Два цілих числа a і b називають одноподільними на ціле число m, якщо або і a, і b діляться на m, або ні a, ні b не діляться на нього.

Позначення

$a \vdots b$ означає, що a ділиться на b, або що число a кратне числу b.

$b \mid a$ або $b \setminus a$ означає, що b ділить a, або, що теж саме: b — дільник a.

Будь-яке ціле число є дільником нуля, при цьому частка дорівнює нулю: $0 \vdots a$. Будь-яке ціле число ділиться на одиницю: $a \vdots 1$.

На нуль ділиться лише нуль: $a \vdots 0 \quad \Rightarrow \quad a = 0$, причому частка в цьому випадку не визначена.

Одиниця ділиться націло лише на одиницю: $1 \vdots a \quad \Rightarrow \quad a = \pm 1$.

Модульна арифметика (modular arithmetic) — це система арифметики цілих чисел, в якій числа «обертаються навколо» деякого значення — модуля. Модульна арифметика оперує з циклічними структурами. Для математика циферблат — це mod 12.

Два цілих числа a і b називаються рівними (конгруентними) за модулем n, якщо при цілочисельному діленні на n вони мають однакові

остачі. Рівність чисел a і b за модулем n записують так: $a \equiv b \pmod{n}$. Це є порівняння чисел a і b. Число m зветься модулем порівняння.

Розв'язування лінійних рівнянь

Лінійне рівняння записується у вигляді $a \cdot x \equiv b \pmod{n}$.

Розв'язок можна отримати безпосередньо діленням $x \equiv \frac{b}{a} \pmod{n}$ або за допомогою формули $x \equiv b \cdot a^{\varphi(n)-1} \pmod{n}$, якщо НСД $(a, n) = 1$, тобто взаємно прості числа. Функція $\varphi(n)$ — функція Ейлера, яка дорівнює кількості натуральних чисел, не більших n і взаємно простих з ним. Якщо НСД $(a, n) \neq 1$, порівняння або має неєдиний розв'язок, або немає розв'язків.

Як легко побачити, порівняння $2 \cdot x \equiv 3 \pmod{4}$ немає розв'язків на множині натуральних чисел.

Інше порівняння $4 \cdot x \equiv 6 \pmod{22}$ має два розв'язки $x = 7,\ x = 18$.

В програмах У **Pascal** та **Basic** оператор **mod** (від modulus) обчислює остачу від ділення, а **div** (від division) обчислює неповну частку:
78 mod 33 = 12; 78 div 33 = 2.

В **C/C++, C#, Java** для цілочисельних операндів оператор / — оператор цілочисельного ділення (ділення націло), відкидається ціла частина.

В Python оператором цілочисельного ділення є //.

У всіх перерахованих вище мовах, крім Pascal, Basic оператором ділення за модулем (залишок від ділення) є %, причому у C#, Java, JavaScript, Python він працює навіть для дійсних чисел, на відміну від модульної арифметики. Наприклад: 7.5 % 1.2 = 0.30000000000000027.

З точки зору від'ємних до особливостей відносяться // та % у **Python**.

Так 17/3 = 5.666666666666667;

7 // 4 = 1; 7 % 4 = 3; -7 // 4 = -2; -7 % 4 = 1.

В **C/C++, C#, Java**

7 / 4 = 1; 7 % 4 = 3; -7 / 4 = -1; -7 % 4 = -3.

В **C/C++**

div(7,4).quot = 1; div(7,4).rem = 3; div(-7,4).quot = -1; div(-7,4).rem = -3.

В **JavaScript** 7 % 4 = 3; -7 % 4 = -3.

В **PHP** intdiv() — цілочисельне ділення; fmod() — вертає дробовий залишок від ділення за модулем;

echo 7 % 4 виводить 3; -7 % 4 — -3; intdiv(-7,4) = -1.

Якщо $n \vdots k$, при діленні натуральних чисел n, k отримуємо $\lfloor n/k \rfloor$ (округлення донизу). Для отримання округлення догори $\lceil n/k \rceil$ можна додати k-1 — (n+k-1)/k, наприклад в Python — (n+59)//60.

Прості числа

Просте число (prime, prime number, простое число) — це натуральне число, яке має рівно два різних натуральних дільники (лише 1 і саме число). Решту чисел, окрім одиниці, називають складеними. Таким чином, всі натуральні числа, більші від одиниці, розбивають на прості і складені.

Послідовність простих чисел $2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, \ldots$ (послідовність A000040 з Онлайн енциклопедії послідовностей цілих чисел, OEIS).

Основна теорема арифметики стверджує, що кожне натуральне число більше одиниці (1), можна представити як добуток простих чисел, причому, в єдиний спосіб з точністю до порядку множників.

Простота одиниці Більшість стародавніх греків навіть не вважали 1 числом, тому вони не могли вважати його простим. До Середніх віків і епохи Відродження багато математики включали 1 в якості першого простого числа. В середині XVIII століття Християн Гольдбах вніс до списку 1 в якості першого простого числа в своїй знаменитій листуванні з Леонардом Ейлером; проте сам Ейлер не рахував 1 простим числом. У XIX столітті багато математики як і раніше вважали число 1 простим числом. Наприклад, список простих чисел Дерріка Нормана Лемера до 10 006 721 числа, перевиданий 1956 році, починався з 1 в якості першого простого числа. Кажуть, що Анрі Лебег є останнім математиком, який назвав 1 простим. До початку XX століття математики стали приходити до консенсусу про те, що 1 не є простим числом, а скоріше формує свою спеціальну категорію — «одиницю».

Простих чисел нескінченно багато Це твердження згадується як теорема Евкліда в честь давньогрецького математика Евкліда, оскільки перше відоме доказ цього твердження приписується йому. Відомо ще багато доказів нескінченності простих чисел, в тому числі аналітичний доказ Ейлера, доказ Гольдбаха на основі чисел Ферма, доказ Фурстенберг з використанням загальної топології і елегантне доказ Куммера.

Перевірка на простоту

Перевірка, чи дане число є простим. Тестом простоти (або перевіркою простоти) (primality test) називається алгоритм, який, прийнявши на вході число, дозволяє або не підтвердити припущення про складові числа, або точно стверджувати його простоту. У другому випадку він

називається істинним тестом простоти. Завдання тесту простоти відноситься до класу складності P, тобто час роботи алгоритмів її рішення залежить від розміру вхідних даних поліноміально, що було доведено в 2002 році.

Існуючі алгоритми перевірки числа на простоту можуть бути розділені на дві категорії: справжні тести простоти і імовірнісні тести простоти. Результатом обчислень справжніх тестів завжди є факт простоти або складені числа. Імовірнісний тест показує, чи є число простим з певною ймовірністю. Числа, що задовольняють імовірнісному тесту простоти, але є складовими, називаються псевдопростими.

До істинних тестів простоти відноситься і перебір дільників.

Створення списку простих чисел

Решето Ератосфена, решето Сундарама та решето Аткіна дають прості способи складання початкового списку простих чисел до певного значення.

Решето Ератосфена — алгоритм знаходження всіх простих чисел, що не перевищують деяке натуральне число n.

Системи числення

Системою числення, або нумерацією (numeral system, system of numeration, система счисления), називається сукупність правил і знаків, за допомогою яких можна відобразити (кодувати) будь-яке невід'ємне число.

Розрізняють наступні типи систем числення: позиційні, непозиційні, змішані.

Позиційні системи числення (позиційна нотація, positional systems) — система числення, в якій значення кожного числового знака (цифри) в запису числа залежить від його позиції (розряду). Таким чином, позиція цифри має вагу у числі. Здебільшого вага кожної позиції кратна деякому натуральному числу b, $b > 1$, яке називається основою системи числення (radix).

У позиційній системі числення з основою b число подають у вигляді лінійної комбінації степенів числа b:

$$x = \sum_{k=0}^{n} a_k b^k,$$

де a_k і k — цілі, $0 \leq |a_k| < |b|$, a_k — цифри числа. Основа позиційної системи числення не обов'язково повинна бути натуральним числом,

узгоджену систему числення можна створити на основі від'ємного цілого числа, або із ірраціональною базою (наприклад на числі e або основі золотого перерізу).

Radix — латинське слово для "корінь". Корінь можна вважати синонімом бази, в арифметичному значенні.

Винахід позиційної системи числення, заснованої на помісному значенні цифр, приписують шумерам і вавилонцям. Її було розвинуто індусами і вона отримала неоціненні наслідки для історії людської цивілізації.

До числа таких систем належить сучасна десяткова система числення (з основою $b = 10$), виникнення якої пов'язують із лічбою на пальцях. У середньовічній Європі вона з'явилася через італійських купців, які у свою чергу запозичили її у мусульман.

Найширше поміж інших використовуються двійкова, десяткова, вісімкова, шістнадцяткова, шістдесяткова.

Шістдесяткова система числення — це позиційна система числення з основою шістдесят. Виникла в шумерів у 3 тисячолітті до н.е., використовувалась у стародавній Вавилонії. Використовується сьогодні в модифікованій формі як основа сучасної кругової системи координат (градуси, хвилини та секунди), географічних координат та вимірювання часу.

Для натуральних чисел остання цифра отримується як залишок від ділення на основу системи числення.

Непозиційні системи числення — системи числення у яких величина, яку позначає цифра, не залежить від позиції її у числі. При цьому система може накладати обмеження на позиції цифр, наприклад, щоб вони були розташовані по спаданню, чи згруповані за значенням.

Типовим прикладом непозиційної системи числення є римська система числення (або римські числа), в якій як цифри використовуються латинські букви. Натуральні числа записуються за допомогою повторення цих цифр. При цьому, якщо більша цифра стоїть перед меншою, то вони додаються (принцип додавання), якщо ж менша перед більшою, то менша віднімається від більшої (принцип віднімання). Останнє правило застосовується тільки для уникнення чотириразового повторення однієї цифри.

Для правильного запису великих чисел римськими цифрами необхідно спочатку записати число тисяч, потім сотні, потім десятків і, нарешті, одиниць.

Деякі з цифр можуть повторюватися, але не більше трьох разів поспіль; таким чином, з їх допомогою можна записати будь-яке ціле число не більше 3999

Існує традиція, що надає перевагу зображенню «4» як «IIII» римськими цифрами, однак до олімпіадних задач таке не відноситься.

В системі римських цифр відсутній нуль.

Римляни для чисел 5000 та більших використовували символи, що не входять до ASCII символів.

Змішана система числення є узагальненням системи числення з основою b і її часто відносять до позиційних систем числення. Основою змішаної системи є послідовність чисел, що зростає, $\{b_k\}_{k=0}^{\infty}$ і кожне число x представляється як лінійна комбінація: $x = \sum_{k=0}^{n} a_k b_k$, де на коефіцієнти a_k (цифри) накладаються деякі обмеження. Якщо $b_k = b^k$ для деякого b, то змішана система збігається з b-основною системою числення.

Найвідомішим прикладом змішаної системи числення є представлення часу у вигляді кількості діб, годин, хвилин і секунд. При цьому величина d днів h годин m хвилин s секунд відповідає значенню $d \cdot 24 \cdot 60 \cdot 60 + h \cdot 60 \cdot 60 + m \cdot 60 + s$ секунд.

Система числення Фібоначчі — змішана система числення для цілих чисел на основі чисел Фібоначчі $F_2 = 1$, $F_3 = 2$, $F_4 = 3$, і т.д.

Будь-яке невід'ємне ціле число a можна єдиним чином подати послідовністю бітів $\ldots\varepsilon_k\ldots\varepsilon_4\varepsilon_3\varepsilon_2$ $(\varepsilon_k \in \{0,1\})$ так що $a = \sum_k \varepsilon_k F_k$, причому послідовність $\{\varepsilon_k\}$ містить лише скінченне число одиниць, і не має пар сусідніх одиниць: $\forall k \geq 2 : (\varepsilon_k = 1) \Rightarrow (\varepsilon_{k+1} = 0)$. За винятком останньої властивості, дане подання аналогічне двійковій системі числення.

Цифри

Цифри (від арабського «сифр» («нуль»)) — знаки, символи для запису чисел. Цифра це єдиний окремий символ (такий як "2"або "5") що використовується самостійно, або у комбінації з іншими (такій як "25"), для представлення чисел відповідно до правил деякої позиційної системи числення.

Індо-арабська або індійська система числення є позиційною десятковою системою числення розроблена у 1—4 століттях індійськими математиками. Цифри виникли в Індії і в 10—13 ст. (можливо не пізніше V століття) були занесені в Європу арабами, через що часто згадуються як «арабські». Сучасне зображення цих цифр — 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 (десяткова система числення).

Двійкова система використовує наступні цифри — 0, 1; вісімкова — 0, 1, 2, 3, 4, 5, 6, 7; шістнадцяткова — 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E,

F (літери заголовні або рядкові); цифри майя : від 0 до 19 (специфічні зображення).

Шістдесяткова система числення окремих цифр немає.

Літери з давен давен використовувались як цифри, в тому числі, на сьогодні, в мовах церковнослов'янська, давньогрецька, іврит.

Алфавітна запис чисел — система, в якій буквам (всім або тільки деяким) приписуються числові значення, часто (але не завжди) такі порядку букв в алфавіті. Найчастіше перші дев'ять букв отримують значення від 1 до 9, наступні дев'ять — від 10 до 90 і т. П. Для запису числа складаються літери, сума значень яких висловлює це число. Для дуже великих чисел застосовуються свого роду діакритичні знаки, які показують, наприклад, що перед нами не одиниці, а тисячі.

Для запису чисел могли застосовуватися як рядкові, так і прописні букви. Порядок: сотні-десятки-одиниці. Числа від слів тексту відрізнялися тим, що над ними проводилася риса і (або) після числа ставилося штрих («числовий апостроф»). Тисячі, десятки тисяч, сотні тисяч позначалися тими ж буквами, що й прості одиниці, десятки, сотні, але зі штрихом внизу зліва.

Кирилична система буква в букву відтворює грецьку. Для запису чисел використовуються майже виключно малі літери. Порядок звичайний: сотні-десятки-одиниці, але в числах, що закінчуються на 11, 12, ..., 19, останні два знаки переставляються згідно слов'янському прочитанню (один-на-дцять, тобто спершу «один», а потім «дцять» = 10). Тисячі, десятки тисяч і сотні тисяч позначаються тими ж буквами, що й звичайні одиниці, десятки, сотні, але зліва (або зліва знизу) ставиться особливий значок. Для відмінності від звичайних слів тексту над числом ставиться особливий знак «Титло» (над єдиною або над передостанньою буквою).

Сьогодні все ще римські цифри — латинські літери: I (1), V (5), X (10), L (50), C (100), D (500), M (1000).

Системи до 36-ї основи (в латиниці 26 літер) після 10-ї основи використовують латинські літери (малі або великі) — в 36-основній системі цифри $0, 1, \ldots, 9, A, B, \ldots, Z$.

Ознаки подільності

Ознака подільності — алгоритм, що дозволяє порівняно швидко визначити, чи є число кратним заздалегідь заданому. Якщо ознака подільності дозволяє з'ясувати не тільки подільність числа на заздалегідь задане, але і залишок від ділення, то його називають ознакою рівноостаточності.

Як правило, ознаки подільності застосовуються при ручному рахунку та для чисел, представлених в конкретній позиційній системі числення (зазвичай десяткової). Поза це, для великих чисел, що не поміщаються в розрядну сітку CPU (64 біти), або для прискорення розрахунків, для визначення подільності не доцільно виконувати безпосередньо ділення (за модулем), а можна скористатись ознаками подільності. Останнє в першу чергу відноситься до мов з вбудованою довгою арифметикою.

На 2 ділиться без остачі ціле число, остання цифра якого парна.

Число ділиться на 3 тоді, коли сума його цифр ділиться на 3.

Число ділиться на 4 тоді, коли його останні дві цифри утворюють число подільне на 4

На 5 ділиться ціле число, остання цифра якого дорівнює 5 або 0

Число n ділиться на 6 тоді і тільки тоді, коли воно $n \vdots 2$ і $n \vdots 3$.

Ділиться на 7:

- якщо потроєна сума десятків разом з одиницями ділиться на 7.
- якщо сума подвоєного числа без останніх двох цифр та числа з двох останніх цифр ділиться на 7.
- числа без останньої цифри, та останньої цифри помноженої на 5, ділиться на 7.

Число ділиться на 8 тоді і тільки тоді, якщо число, утворене його трьома останніми цифрами ділиться на 8.

Число ділиться на 9 тоді і тільки тоді, якщо сума його цифр у десятковому запису ділиться на 9

Ділиться на 10 тоді і тільки тоді, якщо остання його цифра — 0.

На 11:

- число розбивається на блоки по дві цифри, починаючи з кінця. Сума блоків повинна ділитись на 11.
- якщо різниця між числом без останньої цифри і останньою цифрою ділиться на 11.
- якщо сума цифр, що стоять на парних місцях відрізняється від суми цифр, що стоять на непарних місцях, починаючи з кінця, на число, що кратне 11.

Число ділиться на 13:

- коли сума числа десятків з чотирикратною цифрою в розряді одиниць ділиться на 13. Наприклад 845 ділиться на 13, так як на 13 діляться $84 + 5 \cdot 4 = 104$ і $10 + 4 \cdot 4 = 26$.

- коли різниця числа десятків з дев'ятикратним числом, що стоїть в розряді одиниць, ділиться на 13. Наприклад 845 ділиться на 13, так як на 13 діляться $84 - 9 \cdot 5 = 39$.

НСД, НСК

Найбільший спільний дільник (НСД, GCD — greatest common divisor, НОД — наибольший общий делитель) двох або більше невід'ємних чисел — найбільше натуральне число, на яке ці числа діляться без остачі.

Найбільший спільний дільник двох чисел a і b позначається як НСД(a,b), деколи (a,b), НОД(a,b). В англомовній літературі прийнято вживати позначення gcd(a,b).

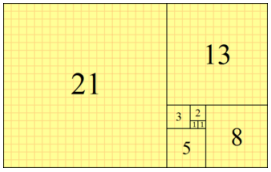
НСД знаходять методом розкладу на прості множники та за алгоритмом Евкліда.

Найменше спільне кратне (НСК, LCM — least common multiple, lowest common multiple, НОК — наименьшее общее кратное) двох цілих чисел — найменше натуральне число, яке є кратним обох цих чисел.

НСК знаходять методом розкладу на прості множники та з НСД.

НСК(a,b) = |ab|/НСД(a,b), в інших позначеннях $\operatorname{lcm}(a,b) = \frac{|a \cdot b|}{\gcd(a,b)}$.

Числа Фібоначчі



Послідовність Фібоначчі, числа Фібоначчі (Fibonacci numbers, числа Фибоначчи) — задана лінійним рекурентним співвідношенням другого порядку

$F_0 = 0, \quad F_1 = 1, \quad F_n = F_{n-1} + F_{n-2},$

$n \geqslant 2, \ n \in \mathbb{Z}.$

Послідовність названа на честь математика XIII століття Леонардо Пізанського (Фібоначчі) (стор.280) з Пізи.

Круглі числа

Круглими числами (Round number) відносно деякої позиційної системи числення називають цілі степені її основи. В цій системі числення такі числа мають вигляд одиниці з наступними нулями. Кількість нулів справа від одиниці дорівнює показнику степені основи.

В десятковій системі числення круглі числа — це $10_{10} = 10^1$, $100_{10} = 10^2$, $1\,000_{10} = 10^3$ тощо.

В двійковій системі числення круглими числами є $10_2 = 2_{10} = 2^1$, $100_2 = 4_{10} = 2^2$, $1\,000_2 = 8_{10} = 2^3$ тощо.

Будь-яке число буде круглим в деякій системі числення. Наприклад, число n буде круглим в системі числення за основою n: $n = 10_n$.

Фігурні числа

Фігурні числа (figurate number, фигурные числа) — це числа, які можна представити у вигляді регулярних дискретних геометричних об'єктів (наприклад, множин кругів чи куль), які щільно виповнюють правильні геометричні фігури. Наприклад, трикутне число — це кількість кругів однакового діаметру з яких можна скласти правильний трикутник. Аналогічно визначають квадратні, п'ятикутні та інші числа. Назва конкретного виду фігурних чисел відображає назву відповідної геметричної фігури. Вважається, що від цих чисел пішов вираз «піднести число до квадрату чи кубу».

Багатокутні числа зустрічаються вже у піфагорійців (VI ст. до н. е.) на думку яких вони відіграють важливу роль у структурі Всесвіту (див. Тетраксис) та у роботах подальших грецьких математиків (Ератосфен, Гіпсикл). Особливо детально їх вивчали математики перших століть нашої ери: Нікомах, Теон Смірнській (II ст.) і їх сучасники. Ними захоплювався і батько грецької алгебри Діофант III–IV ст. н. е.), що написав про них цілу книгу, яка дійшла до нас. Грецькі математики дослідили різні властивості багатокутних чисел, які, зазвичай, доводилися за допомогою геометричних побудов на фігурах.

Незалежно від грецьких математиків багатокутними числами займалися індійські і китайські математики.

Велику увагу фігурним числам приділяли і перші математики середньовічної Європи:Фібоначчі, Пачолі, Кардано та інші. В Новий час багатокутні числа досліджувались Ферма (XVII ст.), Валлісом, Ейлером, Лагранжем (XVIII ст.), Гаусом (XIX ст.) та ін. Ферма сформулював (1637) так звану «золоту теорему» (або теорему Ферма про багатокутні числа):

Довільне натуральне число є сумою щонайбільше n n-кутних чисел, тобто

Довільне натуральне число — або трикутне, або сума двох чи трьох трикутних чисел;

Довільне натуральне число — або квадратне, або сума двох, трьох чи чотирьох квадратних чисел; (Теорема Лагранжа про чотири квадрати);

Довільне натуральне число — або п'ятикутне, або сума від двох до п'яти п'ятикутних чисел; і т. д.

Ферма не міг дати доведення цієї теореми, що слідує, за його словами, це одна «з багатьох глибоко прихованих таємниць чисел». Пройшовши

через руки Ейлера, Лагранжа, Лежандра і Гауса, теорема Ферма була повністю доведена французьким математиком Коші у 1813 році. З цієї теореми випливає багато важливих властивостей чисел.

Зазначимо, що в європейській математиці інколи фігурними числами називалися коефіцієнти членів степенів бінома $(a+b)^n$ при $n = 1, 2, 3, 4, \ldots$ тобто числа з трикутника Паскаля.

У теорії чисел і комбінаторики фігурні числа пов'язані з багатьма іншими класами цілих чисел — біноміальними коефіцієнтами, досконалими числами, числами Мерсенна, Ферма, Фібоначчі, Люка і іншими.

З часів піфагорійців (VI століття до н.е.) традиційно розрізняють такі види фігурних чисел (наприклад, в VII книзі «Начал» Евкліда):

Лінійні числа — числа, які не розкладаються на множники, більше одиниці, тобто це ряд простих чисел, доповнений одиницею (у Евкліда використовується термін «перші числа», $\pi\rho\omega\tau o\iota\ \alpha\rho\iota\theta\mu o\iota$:
$1, 2, 3, 5, 7, 11, 13, \ldots$ (послідовність A008578 в OEIS).

Плоскі числа — числа, представимо у вигляді добутку двох співмножників, більших одиниці, тобто складені: $4, 6, 8, 9, 10, 12, \ldots$ (послідовність A002808 в OEIS).

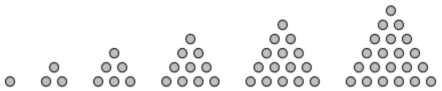
Частковим випадком є прямокутні числа (інколи «продовгуватими»), що є твором двох послідовних цілих чисел, тобто мають вигляд n(n+1).

Тілесні числа — числа, з трьох співмножників: $8, 12, 16, 18, 20, 24, 27, \ldots$ (послідовність A033942 в OEIS).

Багатокутні числа - числа, пов'язані з певним многокутником.

Просторові багатогранні числа - числа, пов'язані з певним многогранником.

Трикутні числа



Трикутне число (triangular/triangle number, треугольное число) — один з типів фігурних чисел, який визначається як число точок, які можуть бути розставлені в формі правильного трикутника (див. малюнок). Очевидно, з чисто арифметичної точки зору, n-е трикутне число — це сума n перших натуральних чисел.

$$T_n = \sum_{k=1}^{n} k = 1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2} = \binom{n+1}{2},$$

де $\binom{n+1}{2}$ — біноміальний коефіцієнт.

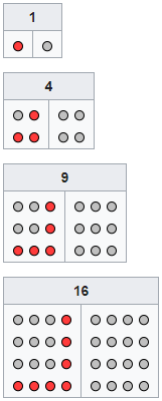
Трикутне число є адитивним варіантом факторіалу.

Сума двох послідовних трикутних чисел — квадратне число, тобто $T_n + T_{n-1} = n^2$.

Кожне парне досконале число є трикутним.

Квадратні числа

Квадрат або **квадратне число** (square number or perfect square; полный квадрат или квадратное число) — додатне ціле число, яке може бути записане у вигляді квадрата деякого іншого числа (інакше кажучи, число, квадратний корінь якого цілий). Геометрично таке число може бути представлене у вигляді площі квадрата з цілочисловою стороною.



$1^2 + 2^2 + 3^2 + \cdots + n^2 = n(n+1)(2n+1)/6$

Кожне число може бути представлене як сума 4 квадратів (Теорема Лагранжа про чотири квадрати).

Остання цифра квадрата (в десятковому записі) може бути лише $0, 1, 4, 5, 6, 9$ (квадратичний лишок по модулю 10).

Дві останні цифри квадрата (в десятковому записі) можуть бути тільки

$00, 01, 04, 09, 16, 21, 24, 25, 29, 36, 41, 44, 49, 56, 61, 64, 69, 76, 81, 84, 89, 96.$

Квадратні числа (і тільки вони) мають непарну кількість дільників.

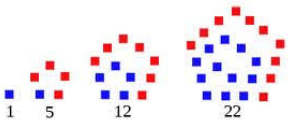
Квадрат не может оканчиваться нечётным количеством нолей.

Квадрат або ділиться на 4, либо при делении на 8 даёт остаток 1.

Квадрат либо делится на 9, либо при делении на 3 даёт остаток 1

П'ятикутні числа

П'ятикутне число (pentagonal number, пятиугольное число) — це фігурне число, яке розширює поняття трикутних і квадратних чисел до п'ятикутника.



n-те п'ятикутне число p_n — це кількість різних точок у шаблоні, що складається з контурів правильних п'ятикутників зі сторонами до n точок, коли п'ятикутники перекриваються так, що вони мають одну спільну вершину.

p_n задається формулою $p_n = \frac{3n^2-n}{2}$.

Послідовність п'ятикутних чисел ($n \geq 1$) (A000326 в OEIS)

$1, 5, 12, 22, 35, 51, 70, 92, 117, 145, 176, 210, \ldots$

Узагальнені п'ятикутні числа отримують із наведеної вище формули, але $n = 0, \pm1, \pm2, \ldots$

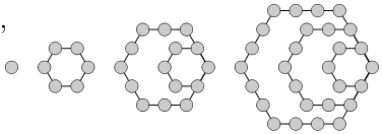
Для заданого додатного натурального числа x, щоб перевірити, чи є це число п'ятикутним (неузагальненим), необхідно обчислити:

$n = \frac{\sqrt{24x+1}+1}{6}$.

Число x є п'ятикутним тоді і тільки тоді, коли n — натуральне число.

Існують також центровані п'ятикутні числа і квадратні п'ятикутні числа.

Шестикутні числа



Шестикутне число (hexagonal number, шестиугольное число) — це фігурне число. n-те шестикутне число h_n — це кількість різних точок у шаблоні, що утворюють контур правильних шестикутників зі сторонами до n точок, коли шестикутники перекриваються так, що вони мають одну спільну вершину.

n-е шестикутне число визначається за допомогою формули

$$h_n = 2n^2 - n = n(2n-1) = \frac{2n(2n-1)}{2}.$$

Послідовність шестикутних чисел (A000384 в OEIS)

$$1, 6, 15, 28, 45, 66, 91, 120, 153, \ldots$$

Кожне шестикутне число — це трикутне число, але лише кожне третє трикутне число (1-е, 3-е, 5-е, 7-е тощо) — це шестикутне число. Як і для трикутного числа, цифровий корінь в основі 10 шестикутного числа може бути лише 1, 3, 6 або 9. Набором цифрових коренів, що повторюється через кожні дев'ять членів, є "1 6 6 1 9 3 1 3 9".

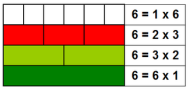
Можна ефективно перевірити, чи є натуральне число x шестикутним числом, за допомогою формули $n = \frac{\sqrt{8x+2}+1}{4}$. Якщо n — натуральне число, то x — n-е шестикутне число.

n-е число шестикутної послідовності можна представити у вигляді суми як

$$h_n = \sum_{i=0}^{n-1} (4i+1).$$

Існують також центровані шестикутні числа та шестикутні квадратні числа.

Досконалі числа



Досконалі числа (perfect number, совершенные числа) — натуральне число, яке дорівнює сумі всіх своїх дільників крім самого числа.

Найменшим досконалим числом є 6: $6 = 3 + 2 + 1$, наступне досконале число — 28: $28 = 14 + 7 + 4 + 2 + 1$.

Досконалі числа утворюють послідовність: 6, 28, 496, 8 128, 33 550 336, 8 589 869 056, 137 438 691 328, 2 305 843 008 139 952 128, 2 658 455 991 569 831 744 654 692 615 953 842 176, 191 561 942 608 236 107 294 793 378 084 303 638 130 997 321 548 169 216, ...

Перші два досконалі числа були відомі ще в глибоку давнину. Наступні два — 496 і 8 128 знайшов в IV столітті до н.е. Евклід і тільки через півтори тисячі років було знайдене ще одне досконале число — 33 550 336. До середини XX століття було знайдено ще 7 таких чисел. Починаючи з 1952 року для пошуку досконалих чисел почали застосовувати ЕОМ і якщо перше досконале число (6) однозначне, то 24-те має понад 12 000 знаків.

Евклід не тільки відшукав два досконалих числа, а і дав ключ до пошуку парних досконалих чисел. Він довів досконаліть чисел які можна представити у вигляді $2^{p-1}(2^p - 1)$, де $2^p - 1$ — просте число (для того щоб число $2^p - 1$ було простим необхідно, але недостатньо, щоб p було простим). Усі відомі досі досконалі числа парні, проте не існує доказу того, що непарних досконалих чисел немає.

В XVII столітті досконалі числа шукав французький математик Мерсенн. Він припустив, що при p=17, 19, 31, 67, 127 і 257 формула Евкліда дає досконале число. Проте перевірити своє припущення не зумів через складність обчислень. Правоту Мерсенна для p=17, 19 і 31 довів в XVIII столітті Леонард Ейлер. Пізніше виявилась помилковість передбачень для p=67 і 257, що не заважає називати числа вигляду $2^p - 1$ числами Мерсенна.

На грудень 2018 р. знайдено 51 простих чисел Мерсенна і відповідних до них парних досконалих чисел. Обчисленням нових простих чисел Мерсенна займається проект розподілених обчислень GIMPS.

Факторизація

Факторизація або розкладання на множники — це декомпозиція об'єкту (наприклад, числа, оператора, многочлена або матриці) у добуток інших об'єктів, або множників, добуток декількох операторів нижчого порядку, які після перемноження дадуть вихідний об'єкт.

Факторизація цілого числа — розкладання заданого числа на прості множники.

Найбільш тривіальним алгоритмом факторизації чисел є повний перебір можливих дільників. Складність цього алгоритму дорівнює $O(N^{1/2})$.

2.1.2. Раціональні числа

Множина раціональних чисел є підмножиною алгебраїчних та дійсних чисел.

Раціональні числа — множина раціональних чисел $\mathbb{Q}$ визначається як множина нескоротних дробів із цілим чисельником і натуральним знаменником: $\mathbb{Q} = \left\{ \frac{m}{n}, m \in \mathbb{Z}, n \in \mathbb{N} \right\}$

В Python вбудована обробка дробів.

2.1.3. Дійсні числа

Дійсні числа — елементи числової системи, яка містить у собі раціональні числа і, в свою чергу, є підмножиною комплексних чисел. Математична абстракція, яка виникла з потреб вимірювання геометричних і фізичних величин навколишнього світу, а також виконання таких математичних операцій як добування кореня, обчислення логарифмів, розв'язування алгебраїчних рівнянь.

Кожному дійсному числу на числовій прямій можна поставити у відповідність єдину точку, і навпаки, кожна точка представлятиме єдине дійсне число.

Множину дійсних чисел стандартно позначають $\mathbb{R}$ чи $\mathbf{R}$. Дійсне число подають в вигляді з фіксованою або плаваючою комою/крапкою.

Числа можуть бути в вигляді десяткового дробу та експоненційному поданні — $N = M \cdot n^p$, де N — число; M — мантиса; n — основа (показникової функції); p (ціле) — порядок (числа).

Необхідно зауважити, що незважаючи на назву типів чисел, в цифровій техніці принципово дійсні числа апроксимуються раціональними, що частенько може приводити до неточностей розрахунків, тому, як описано вище, якщо можна знайти розв'язок задачі в обхід дійсних чисел, потрібно не виходити за межі цілих чисел.

Арифметична прогресія

Арифметична прогресія це послідовність дійсних чисел (членів прогресії) виду

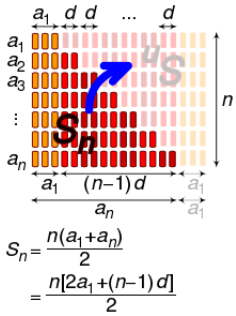
$$a_1, a_2, ..., a_n, ... = a_1, a_1 + d, a_1 + 2d, ..., a_1 + (n-1)d, ...$$

де a_1 — перший член прогресії, d — крок або різниця прогресії.

Довільний n-й член прогресії визначається $a_n = a_1 + (n-1)d, \forall n \geq 1$.

Сума перших n членів арифметичної прогресії може бути виражена

$$S_n = \sum_{i=1}^{n} a_i = \frac{a_1 + a_n}{2} n = \frac{2a_1 + d(n-1)}{2} n.$$



Сума перших n натуральних чисел:

$$1 + 2 + ... + n = \frac{n(n+1)}{2}$$

є трикутним число.

2.1.4. Комплексні числа

Комплексні числа — розширення поля дійсних чисел, зазвичай позначається $\mathbb{C}$. Будь-яке комплексне число може бути представлене як формальна сума $x + iy$, де x і y — дійсні числа, i — уявна одиниця.

В математиці, фізиці, інженерній справі уявна одиниця позначається як латинська i (ввів Ейлер від imaginary) або j. В Python використовується j.

2.2. Рядки

Рядок (string) — це тип даних, значеннями якого є довільна послідовність символів. В рядок можуть входити літери та спеціальні символи.

Рядок є масивом символів у більшості мов програмування. Доступ до окремих символів — за індексом (номером в рядку).

Символи в різних кодуваннях займають різну кількість байт. В олімпіадних задачах використовують тільки кодування ASCII, в класичному розумінні, тільки латиницю та спецсимволи.

В Python list() розкладає рядок на список окремих символів. Так, наприклад, list('Жмурко Тетяна') дає ['Ж', 'м', 'у', 'р', 'к', 'о', ' ', 'Т', 'е', 'т', 'я', 'н', 'а']. Така поведінка дозволяє легко розчіпити число на цифри для подальшої обробки, тобто вести обробку чисел як тексту.

2.3. Масиви

Масив (array) — сукупність елементів, впорядкованих за індексами, які зазвичай репрезентовані натуральними числами, що визначають положення елемента в масиві.

В інформатиці індексування масиву починається з 0 в таких мовах програмування, як C, C++, Java, Python та інші.

В Pascal індекс масиву може починатись з довільного числа.

В C/C++, C#, Java масив — набір елементів однакового типу.

В Python, JavaScript, PHP в масиві можуть зберігатись елементи довільних типів.

В C/C++, C#, Java обробка масиву, в тому числі введення-виведення, ведеться тільки поелементно.

В C/C++ ініціалізація —

тип ім'яМасиву[розмір]={списокЗначень};

Якщо список значень коротший за масив, решта заповнюється нулями, тому для заповнення масиву нулями достатньо задати тільки перший елемент — ={0}.

В Pascal описаний масив автоматично заповнюється, як і інші типи, нулевими значеннями, для числового типу — 0.

В Java, аналогічно, після створення масиву за допомогою new, в його комірки записані значення за замовчуванням. Для чисельних типів це 0, для boolean — false.

В Python масив задається списком (list). Кількість елементів масиву несуттєва, її можна не вводити, на відміну від інших мов програмування. Для масиву працюють функції max, min, sum.

2.3.1. Одновимірні масиви

Доступ до i-го елементу масиву a — $a[i]$.

В Java метод Arrays.fill() дозволяю заповнити масив однаковими значеннями. Сортування масиву a виконується методами Arrays.sort(a) і Arrays.sort(a,index1,index2).

В Python Сортування масиву здійснюється методом a.sort() або функцією sorted(a).

Зручним є використання масиву як масиву відповідей. Якщо кількість результатів невелика доцільно розрахувати результати окремо, а відповіді занести в масив і ним користуватись в програмі-розв'язок. Тут проявляється «Золоте правило» в програмуванні — виграємо в часі (розрахунок) або пам'яті (результати в таблицях, масивах).

В Python можна використовувати довільні UTF символи, наприклад, ©$\hbar^2$ (U+00A9, U+045B, U+00B2), в тому числі і в масивах.

Розглянемо деякий масив

```
a = ['zhm', 7,1.3e18,2>0, 2>=3,[3,5,8], \
    (13,21),"copyright",10000000000000000000.]
```

Виведення масиву — print(a)

['zhm', 7, 1.3e+18, True, False, [3, 5, 8], (13, 21), 'copyright', 1e+20]

Виведення масиву рядків без дужок

```
r=['one','second',"3","prime"]
print(' '.join(r))
```

Виведення списку Python без дужок — print(*a). В прикладі маємо zhm 7 1.3e+18 True False [3, 5, 8] (13, 21) copyright 1e+20

2.3.2. Двовимірні масиви

Доступ до i,j-го елементу масиву a в C/C++, C#, Java, JavaScript, Python, PHP — $a[i][j]$. Доступ до i,j-го елементу масиву a в Pascal — $a[i,j]$.

Користь і проблеми можуть принести зубчасті масиви (jagged array) в C#, JavaScript, Python, PHP. Зубчастий масив — масив, рядки якого (підмасиви) мають різну довжину.

2.4. Матриці

Матриця — математичний об'єкт, записаний у вигляді двовимірними прямокутної таблиці чисел (чи елементів кільця), він допускає операції (додавання, віднімання, множення та множення на скаляр).

Матрицею розміру $m \times n$ (m-на-n, або mn-матрицею) називається множина з mn елементів $a_{i,j}$, розміщених у вигляді прямокутної таблиці з m рядків і n стовпців, а m і n — її розмірністю.

Додавання та віднімання матриць $A \pm B$: $c_{i,j} = a_{i,j} \pm b_{i,j}$.

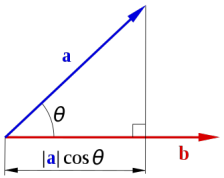
Множення на скаляр αA: $c_{i,j} = \alpha \cdot a_{i,j}$.

Множення матриць $A \times B$: $c_{i,k} = a_{i,j} \times b_{j,k}$. Існує окрема операція поелементного множення матриць.

Обернена матриця A^{-1} задовільняє рівнянню $A^{-1} \times A = A \times A^{-1} = 1$.

2.5. Вектори

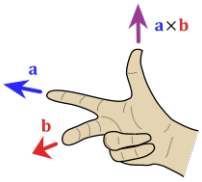
Скалярний добуток — (dot product, scalar product, скалярное произведение) — бінарна операція над векторами, результатом якої є скаляр.



Скалярний добуток геометричних векторів $\vec{x}$ та $\vec{y}$ обчислюється за формулою: $\vec{x} \cdot \vec{y} =$

$|\vec{x}|\,|\vec{y}|\cos\measuredangle(\vec{x},\vec{y})$ де $|\vec{x}|$ та $|\vec{y}|$ є довжинами векторів, а $\cos\measuredangle(\vec{x},\vec{y})$ дорівнює косинусу кута між цими векторами. Як і у випадку звичайного множення, знак множення можна не писати: $\vec{x}\cdot\vec{y}=\vec{x}\vec{y}$.

Векторний добуток — білінійна, антисиметрична операція на векторах у тривимірному просторі. На відміну від скалярного добутку векторів евклідового простору, результатом векторного добутку є вектор (його також називають «векторним добутком»), а не скаляр.



Векторний добуток двох векторів у тривимірному евклідовому просторі — вектор, перпендикулярний до обох вихідних векторів, довжина якого дорівнює площі паралелограма, утвореного вихідними векторами, а вибір з двох напрямків визначається так, щоб трійка з векторів-множників, узятих в такому ж порядку, як записано в добутку, і отриманого вектора була правою. Векторний добуток колінеарних векторів (зокрема, якщо хоча б один з множників — нульовий вектор) вважається рівним нульовому вектору.

Векторний добуток було введено У. Гамільтоном у 1846 році.

Найчастіше для позначення векторного добутку вживається символ $\times$. Векторний добуток позначається також квадратними дужками, в яких співмножники розділені комами. Крім того, в фізичних текстах заведено позначати вектори жирним шрифтом.

$$\vec{u}\times\vec{v}=[\vec{u},\vec{v}]=[\vec{u}\times\vec{v}]=\mathbf{u}\times\mathbf{v}=[\mathbf{u},\mathbf{v}]$$

Довільний вектор в $\mathbb{R}^3$ описується своїми координатами відносно стандартного базису $\{\vec{i},\vec{j},\vec{k}\}$. Векторним добутком двох 3-векторів $\vec{u}=u_1\vec{i}+u_2\vec{j}+u_3\vec{k}$, $\quad\vec{v}=v_1\vec{i}+v_2\vec{j}+v_3\vec{k}$, називається 3-вектор

$\vec{u}\times\vec{v}=(u_2v_3-u_3v_2)\vec{i}+(u_3v_1-u_1v_3)\vec{j}+(u_1v_2-u_2v_1)\vec{k}$, який також символічно записується у вигляді 3×3 детермінанту:

$$\vec{u}\times\vec{v}=\begin{vmatrix}\vec{i}&\vec{j}&\vec{k}\\u_1&u_2&u_3\\v_1&v_2&v_3\end{vmatrix}.$$

$|\vec{u}\times\vec{v}|=|\vec{u}||\vec{v}|\sin\theta$, де θ — це кут між $\vec{u}$ та $\vec{v}$ (довжина векторного добутку або правило паралелограму, площа на векторах $\vec{u}$ та $\vec{v}$).

2.6. Геометрія

2.6.1. Планіметрія

Трикутник

Теорема Піфагора У прямокутному трикутнику сума квадратів катетів дорівнює квадрату гіпотенузи.

Формула Герона 3.21.7 Герон[29] дозволяє визначити площу трикутника S за даними довжинами його сторін a, b і c.

$S = \sqrt{p(p-a)(p-b)(p-c)}$, де $p = \frac{a+b+c}{2}$, — половина периметру трикутника.

Теорема косинусів Для плоского трикутника зі сторонами a, b, c з кутом α, протилежним стороні a, справедливо співвідношення:

$a^2 = b^2 + c^2 - 2bc\cos\alpha$.

Якщо квадрат деякої сторони трикутника більший від суми квадратів двох інших сторін, то протилежний йому кут є тупим:

$a^2 < b^2 + c^2$ або $b^2 + c^2 - a^2 > 0$, то α — гострий. Сума квадратів діагоналей паралелограма дорівнює сумі квадратів його сторін $a^2 > b^2 + c^2$ або $b^2 + c^2 - a^2 < 0$, то α — тупий. Якщо квадрат деякої сторони трикутника дорівнює сумі квадратів двох інших сторін, то протилежний йому кут є прямим:

$a^2 = b^2 + c^2$ або $b^2 + c^2 - a^2 = 0$, то α — прямий.

Доведення теореми косинусів геометричне або з використанням векторів.

2.6.2. Константа Архімеда, число π

Про число π наводимо в тексті тут тільки відповідно до визначення числа. Число π — математична константа, що визначається у Евклідовій геометрії як відношення довжини кола l до його діаметра d.

Вперше позначенням цього числа грецькою літерою π скористався британський (валлійський) математик Вільям Джонс (1706), а загальноприйнятим воно стало після робіт Леонарда Ейлера (1737). Це позначення походить від початкової букви грецьких слів $\pi\varepsilon\rho\iota\varphi\varepsilon\rho\iota\alpha$ — оточення, периферія та $\pi\varepsilon\rho\iota\mu\varepsilon\tau o\zeta$ — периметр.

Оскільки π є ірраціональним числом, його не можна виразити дробом (або що те саме, його десяткове представлення є нескінченним та неперіодичним).

Також π є трансцендентним числом — тобто не є коренем жодного ненульового поліному з раціональними коефіцієнтами.

Стародавні цивілізації користувалися приблизним значенням числа π у практичних цілях. У V столітті н.е. китайські математики за допомогою геометричних методів обчислювали його до сьомого знаку після коми, а індійські — до п'ятого.

Практично, фізикам потрібно тільки 39 цифр числа π, щоб зробити коло розміром як видимий всесвіт з точністю до розміру атома водню.

На початку 20-го століття індійський математик Срініваса Рамануджан відкрив багато нових формул для числа π, деякі з них стали знамениті через свою елегантність та математичну глибину.

Число π в програмах — див.2.9.

2.7. Дати та час

Нульового року в юліанському й григоріанському календарі не було.

2.8. Введення-виведення в програмах

Вхідні дані не потрібно перевіряти на правильність, відповідність умовам задачі. Правильність даних та їх послідовність гарантується.

Принципово не потрібно тексту запрошень-пояснень (інтерфейс користувача), тільки необхідне введення-виведення, задане в умові задачі або прикладах до неї.

Введення-виведення через файли

На C

```
#include <stdio.h>
long a,b;
int main(){
  freopen("input.txt","r",stdin);
  freopen("output.txt","w",stdout);
  scanf("%ld%ld",&a,&b);
  printf("%ld",a+b);
}
```

Pascal

```
var a, b : longint;
begin
  assign(input, 'input.txt'); reset(input);
  assign(output,'output.txt');rewrite(output);
  read(a, b);
  write(a + b);
end.
```

на Java

```
import java.util.*;
import java.io.*;
public class Main{
  public static void main(String[] argv)
      throws IOException{
    new Main().run();
  }
  PrintWriter pw;
  Scanner sc;
  public void run() throws IOException{
    sc = new Scanner(new File("input.txt"));
    int a=sc.nextInt(), b=sc.nextInt();
    pw = new PrintWriter(new File("output.txt"));
    pw.print(a+b);
    pw.close();
  }
}
```

Basic

```
Sub Main()
  open "input.txt" for input as #1
  open "output.txt" for output as #2
  input #1,a#,b#
  print #2,a#+b#
  close #1
  close #2
End Sub
```

Python

```
i,o=open('input.txt'),open('output.txt','w')
```

```
for line in i: o.write(line)
```

На e-olymp немає необхідності ставити позначку про використання файлів, визначення відбувається автоматично. Також допустимо відкрити файл введення, а для введення-виведення використовувати консоль.

Введення-виведення через консоль

Класична пробна задача A+B [30] для перевірки готовності перед олімпіадою, ознайомлення учасників з тестуючою системою:

на C

```
#include <stdio.h>
void main(){
    int a, b;
    scanf("%d%d", &a, &b);
    printf("%d\n", a + b);
}
```

на C++

```
#include <iostream>
using namespace std;
int main(){
    int a, b;
    cin >> a >> b;
    cout << a + b;
}
```

на C#

```
using System;
class Program {
  static void Main() {
    string[] input = Console.ReadLine().Split('␣');
    Console.WriteLine(int.Parse(input[0])+
      int.Parse(input[1]));
  }
}
```

на Java

```
import java.util.Scanner;
public class AplusB {
```

```
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    int a = scanner.nextInt();
    int b = scanner.nextInt();
    System.out.println(a + b);
  }
}
```

на Free Pascal

```
var a, b : longint;
begin
    Read(a, b);
    WriteLn(a + b);
end.
```

на Python

```
a, b = map(int, input().split(' '))
print(a + b)
```

на Haskell

```
main=print.sum.map read.words =<< getLine
```

Для читання невизначеної кількості вхідних даних у Pascal та Python необхідно використовувати файл введення і читати до кінця файлу. В інших мовах програмування також можна використовувати файли.

В C++ можна використати цикл $while(cin)$ або $while(cin >> x)$, де x — змінна, що вводиться, тобто сумістити введення з перевіркою.

2.9. Програмування, оформлення коду

Коди олімпіадних задач можуть використовувати те, що в індустріальному програмуванні є забороненим.

Коментарі. Із-за обмеженого кола користувачів коду — членів команди та обмеженого часу на олімпіаді, в програмах не використовують коментарі. По можливості розв'язок повен проглядатись в коді програми (програма «легко читається»).

Змінні називають короткими іменами (ідентифікаторами). Назви змінних повинні збігатись з вказаними в умові задачі.

Змінну, що використовується тільки один раз, належить опускати, тобто розрахунок робити по місцю змінної.

Парність числа x — $x\%2$ — 1 непарне, 0 парне

Два числа x, y мають однакову парність — $(x + y)\% == 0$.

Однаковий знак чисел Два числа x, y мають однаковий знак: $x \cdot y > 0$.

Одно з чисел 0 — добуток чисел дорівнює нулю.

Заголовний файл (header file) в C/C++. Якщо необхідно використовувати більше ніж заголовний файл введення-виведення, замість їх перерахунку, вказують #include <bits/stdc++.h>, що містить всі необхідні компоненти. Доцільно описувати одразу стандартний простір імен — using namespace std;

Константи в програмах. В ряді програм можуть знадобитись константи π, e. Їх не вводять, а користуються вбудованими.

В Python — math.pi, math.e (import math).

В C/C++ — M_PI, C++ #include <cmath>, C #include <math.h>.

В C#, Java, JavaScript — Math.PI, Math.E.

В Pascal та PHP π є функцією без параметрів — pi().

Число e в програмах може бути розраховане як exp(1).

Бібліотеки та модулі. В Python можна використовувати тільки модулі зі стандартної бібліотеки, наприклад, math, cmath, datetime, decimal, fractions.

Функції. Не використовують власні функції, якщо немає обґрунтованої, нагальної потреби. Належить використовувати вбудовані функції, вони скорочують код та можуть бути ефективніше їх аналогів (реалізації) в коді, наприклад, max() і min()(Pascal, C/C++, Python); swap() (C/C++), sum()(Python) тощо.

Треба зауважити, що в Pascal функції max(), min() мають тільки два числових аргументи, в C++ — декілька, в Python можуть бути числа, списки (масиви). Аналогічно з аргументами в Python у функції sum().

В програмах між log, lg та ln є плутанина. Як в англомовних позначеннях, функція натурального логарифму — log(), а десяткового — log10().

Вбудовані функції bin(), oct(), hex() в Python дозволяють перетворювати в двійкову, вісімкову, шістнадцяткову систему числення, int() — працювати з довільною (основа ≤ 36).

Небезпечна функція eval(), може бути використана в олімпіадному коді.

ООП використовувати недоцільно, за винятком вбудованих методів (неявне використання).

Операторні дужки. Опускаємо операторні дужки **{ }**, **begin end**, якщо в них знаходиться тільки один оператор.

Число рядків має бути мінімальним. Є сенс записувати в один рядок короткі цикли або умови, якщо це логічно обґрунтовано.

В Python, як і в Pascal, ";"(крапка з комою) є подільником між операторами, в Python необов'язковим (використовуються відступи), в Pascal — обов'язковим.

Там де це обгунтовано, в один рядок можна розташовувати декілька коротких операторів.

Файли за потреби відкривають і можна не закривати, це зробить система.

Аналогічно можна опускати закриваючий тег в PHP.

Ініціалізація, початкові значення. В Pascal неініціалізовані змінні містять нулеві значення (0, 0.0, false,).

Для масивів приклад ініціалізації: prime=[True]*N (Python).

Завершення, вихід з програм В C++ програми завершуються return 0; Такий оператор інформує операційну систему про нормальне, безаварійне завершення програми. Отож це можна просто опускати.

Аналогічну функцію виконує exit() в Python. Параметром цієї функції є номер помилки, без параметру — нормальне завершення. При використанні цієї функції параметр опускаємо.

Схожа ситуація в Pascal.

В лістингу програм зустрічаються пробіли (в рядках). Їх вигляд наведено в лапках

```
'␣' "␣"
```

В Python за правилами (PEP 8) рекомендується не більше 79 символів в рядку. Для роздруківок, що наводяться тут, приходиться використовувати символ розбиття рядка коду \.

2.10. Алгоритми

2.10.1. Решето Ератосфена

Решето Ератосфена (sieve of Eratosthenes, решето Эратосфена) — простий стародавній алгоритм знаходження всіх простих чисел менших деякого цілого числа n, що був створений давньогрецьким математиком Ератосфеном (стор.276).

Якщо потрібно знайти всі прості числа менші за певне число N, виписуються всі числа від 1 до N.

1. Перше просте число — два. Викреслимо всі числа більші двох, які діляться на два;
2. Наступне незакреслене число є простим. Викреслимо всі числа більші нього та кратні йому.
3. Повторюємо попередню операцію поки не буде досягнуто число N.

Числа, які залишилися незакресленими після цієї процедури — прості.

Оцінка складності Алгоритм потребує $O(N)$ біт пам'яті та $O(N \log\log N))$ математичних операцій.

На псевдокоді (з модифікацією від i^2):
algorithm Sieve of Eratosthenes:
input: ціле $n > 1$.
output: всі прості до n.
нехай p — булевий масив, що індексує цілі від 2 до n,
початково всі встановлюються в true.
for $i = 2, 3, 4, \ldots$, не більше $\sqrt{n}$ do
if p[i] є true
for $j = i^2, i^2 + i, i^2 + 2i, i^2 + 3i, \ldots$, не більше n do
p[j] := false
return всі i такі що p[i] є true.

Починати можна з i^2, оскільки $i \times 2, i \times 3, \ldots, i \times (i-1)$ вже відмічені (викреслені) попередніми множниками $2, 3, \ldots, i-1$.

На Python це може виглядати так

```
n=1000000
p=[True]*n
for i in range(2,int(n**.5)+1):
    if p[i]:
        for j in range(i**2,n+1,i):
            p[j]=False
```

2.10.2. Алгоритм Евкліда

Алгоритм Евкліда (або евклідів алгоритм) — ефективний метод обчислення найбільшого спільного дільника (НСД). Названий на честь грецького математика Евкліда (стор.274), котрий описав його в книгах VII та X Начал.

Алгоритм Евкліда заснований на тому, що НСД не змінюється, якщо від більшого числа відняти менше. Оскільки більше з двох чисел постійно зменшується, повторне виконання цього кроку дає все менші числа, поки одне з них не дорівнюватиме нулю. Коли одне з чисел дорівнюватиме нулю, те, що залишилось, і є НСД.

Найдавніший опис алгоритму знаходиться в Началах Евкліда (біля 300 до н.е.), що робить його найдавнішим чисельним алгоритмом, яким користуються і нині. Оригінальний варіант алгоритму описував роботу лише з натуральними числами та геометричними довжинами (дійсними числами), алгоритм було узагальнено в XIX столітті на роботу з іншими типами чисел, такими як Гаусові числа та поліноми з однією змінною.

Алгоритм Евкліда ефективно обчислює НСД великих чисел, оскільки виконує операцій не більше, ніж вп'ятеро більше кількості цифр меншого числа (в десятковій системі). Цю властивість було доведено Ґабріелем Ламе (Gabriel Lame) в 1844 році, що позначило початок теорії складності обчислень. Методи підвищення ефективності алгоритму були розроблені в XX столітті.

2.10.3. Перебір дільників

Перебір дільників (trial division, перебор делителей, пробное деление) — алгоритм факторизації або тестування простоти числа шляхом повного перебору всіх можливих потенційних дільників.

Зазвичай перебір дільників полягає в переборі всіх цілих (як варіант простих) чисел від 2 до квадратного кореня з факторизуємого числа n і в обчисленні залишку від ділення n на кожне з цих чисел. Якщо

залишок від ділення на деяке число i дорівнює 0, то i є дільником n. У цьому випадку або n оголошується складеним, і алгоритм закінчує роботу (якщо тестується простота n), або n скорочується на i і процедура повторюється (якщо здійснюється факторизация n). Після досягнення квадратного кореня з n і неможливості скоротити n ні на одне з менших чисел n оголошується простим.

Для прискорення перебору часто вже не перевіряються парні подільники, крім числа 2, а також дільники, кратні трьом, крім числа 3. При цьому тест прискорюється в три рази, так як з кожних шести послідовних потенційних подільників необхідно перевірити тільки два, а саме виду $6 \times k \pm 1$, де k — натуральне число.

В найгіршому випадку перебір доведеться проводити від 2 до $\sqrt{n}$. Складність даного алгоритму $O(n^{1/2})$.

У практичних завданнях даний алгоритм застосовується рідко через його велику обчислювальну складність, однак його застосування виправдане в разі, якщо перевіряються числа відносно невеликі, так як даний алгоритм досить легко реалізуємо, в тому числі в олімпіадних задачах.

2.10.4. Жадібний алгоритм

Жадібний алгоритм (greedy algorithm, жадный алгоритм) — простий і прямолінійний евристичний алгоритм, який приймає найкраще рішення, виходячи з наявних на кожному етапі даних, не зважаючи на можливі наслідки, в надії отримати оптимальний розв'язок. Легкий в реалізації і часто дуже ефективний за часом виконання. Багато задач не можуть бути розв'язані за його допомогою.

Зазвичай, жадібний алгоритм базується на п'яти принципах:

1. Набір можливих варіантів, з яких робиться вибір;
2. Функція вибору, за допомогою якої знаходиться найкращий варіант, який буде додано до розв'язку;
3. Функція придатності, яка визначає придатність отриманого набору;
4. Функція мети, передає значення розв'язку або частковому розв'язку;
5. Функція розв'язку, яка вказує на те, що кінцевий розв'язок знайдено.

Придатний набір варіантів — такий, що обіцяє не просто отримання розв'язку, а отримання оптимального розв'язку задачі.

На відміну від динамічного програмування, за якого задача розв'язується знизу догори, за жадібної стратегії це робиться згори донизу, шляхом здійснення одного жадібного вибору за іншим, зведенням великої задачі до малої.

Жадібний алгоритм добре розв'язує деякі задачі, а інші — ні. Більшість задач, для яких він спрацьовує добре, мають дві властивості: по-перше, до них можливо застосувати принцип жадібного вибору, по-друге, вони мають властивість оптимальної підструктури.

Жадібні алгоритми можна характеризувати як «короткозорі» і «невідновлювані». Вони ідеальні лише для задач з «оптимальною підструктурою». Попри це, жадібні алгоритми найкраще підходять для простих задач. Для багатьох інших задач жадібні алгоритми зазнають невдачі у продукуванні оптимального розв'язку, і можуть навіть видати найгірший з можливих розв'язків.

Розмін монет — видати суму S найменшою можливою кількістю монет. Жадібний алгоритм розв'язання цієї задачі наступний. Беремо найбільшу можливу кількість монет найбільшої вартості a_n: $x_n = \lfloor S/a_n \rfloor$. Аналогічно знаходимо скільки потрібно монет меншого номіналу.Для цієї задачі жадібний алгоритм не завжди дає правильне розв'язок. Проте, на всіх реальних монетних системах жадібний алгоритм видає правильну відповідь.

Вибір заявок На конференції, щоб відвести більше часу на неформальне спілкування, різні секції рознесли по різних аудиторіях. Вчений із надзвичайно широкими інтересами бажає відвідати декілька доповідей у різних секціях. Відомі початок s_i і кінець f_i кожної доповіді. Визначити, яку найбільшу кількість доповідей можна відвідати.

Жадібний алгоритм розв'язку задачі. Вважатимемо, що заявки впорядковано за зростанням часу закінчення. Якщо це не так, то можна відсортувати їх за час $O(n \log n)$.

```
Activity-Selector(s,f)
n ← length[s]
A ← {1}
j ← 1
for i ← 2 to n
   do if s_i ≥ f_j
      then A ← A ∪ i
           j ← i
return A
```

На вхід алгоритму подаються масиви початків і закінчень доповідей. Множина складається з номерів вибраних заявок, а j — номер останньої заявки. Жадібний алгоритм шукає заявку, що починається не раніше від закінчення j-ї, потім знайдену заявку включає в A, а j присвоює її номер.

Алгоритм працює за $O(n \log n)$, тобто за складність сортування, оскільки вибірка має меншу складність $O(n)$. На кожному кроці вибирається найкращий розв'язок.

2.10.5. Схема Горнера

Схема Горнера (правило, метод Горнера, Horner's rule, method, scheme) — алгоритм обчислення значення многочлену, записаного у вигляді суми одночленів, при заданому значенні змінної.

Метод Горнера також дозволяє знайти корені многочлену, а також обчислити похідні поліному в заданій точці. Схема Горнера також є простим алгоритмом для ділення многочлена на біном у вигляді $x-c$. Метод названий на честь Вільяма Джорджа Горнера (стор.273).

Дано многочлен $P(x)$:

$P(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \ldots + a_n x^n, \quad a_i \in \mathbb{R}.$

Обчислимо значення даного многочлена при фіксованому значенні x. Подамо многочлен $P(x)$ в вигляді:

$P(x) = (\ldots(a_n x + a_{n-1})x + a_{n-2})x + \ldots + a_1)x + a_0.$

Починаючи з внутрішніх дужок, послідовно розраховуємо значення двочленів (в поточних дужках). Останній — значення поліному.

2.10.6. Рекурсія

Відомий рекурсивний жарт: «Для того, щоб навчитися програмувати потрібно програмувати».

Рекурсія (recursion) — метод визначення класу чи об'єкту через попереднє задання одного чи декількох (зазвичай простих) його базових випадків чи методів, а потім заданням на їхній основі правила побудови класу, який визначається. Рекурсія — часткове визначення об'єкта через себе, визначення об'єкта з використанням раніше визначених. Рекурсія використовується, коли можна виділити самоподібність задачі.

Рекурсія — виклик функції чи процедури з неї самої (звичайно з іншими значеннями вхідних параметрів) безпосередньо чи через інші функції (наприклад, функція A викликає функцію B, а функція B — функцію A). Кількість вкладених викликів функції чи процедури називається глибиною рекурсії. Варто уникати надлишкової глибини рекур-

сії. Характерна максимальна глибина рекурсії — 100. Велика глибина рекурсії може викликати runtime помилки, бо може викликати переповнення стека викликів.

Рекурсія може бути використана, або проявляється — факторіали, числа Фібоначчі, геометричні фрактали, Ханойські вежі, народній казці-пісні «У попа була собака...», «Дім, що збудував Джек».

Розділ 3

Приклади задач

3.1. Базові можливості мов

3.1.1. e4716 Поділ яблук – 1

n школярів ділять **k** яблук порівну, залишок, що не ділиться, залишається у кошику. Скільки яблук дістанеться кожному школяру?

Вхідні дані

Два додатніх цілих числа **n** та **k**, які не перевищують 1500 — рідко у школі буває більше учнів, да й багато яблук також їсти шкідливо...

Вихідні дані

Вивести кількість яблук, яке дістанеться кожному школяру.

Складність: 2% — 17732/9546/7692/7396.

Тут маємо ділення націло.

На **Python**

```
n,k =int(input()),int(input())
print(k // n)
```

На **Java** (116 ms, 23.7 MiB)

```
import java.util.Scanner;
public class e4716 {
        public static void main(String[] args) {
                Scanner in = new Scanner(System.in);
                int n = in.nextInt();
                int k = in.nextInt();
                System.out.println(k / n);
        }
}
```

3.1.2. e4717 Поділ яблук – 2

n школярів ділять **k** яблук порівну. Залишок, що не ділиться, залишається у кошику. Скільки яблук залишиться у кошику?

Вхідні дані

Два додатніх цілих числа **n** та **k**, не більших за 1500 — рідко у школі буває більше учнів, да й де знайти такий кошик?

Вихідні дані Вивести кількість яблук, що залишаться у кошику.

Складність: 2% — 17732/9546/7692/7396.

Тут маємо залишок від ділення націло.

На **Python**

```
n,k =int(input()),int(input())
print(k % n)
```

На **Java** (119 ms, 23.8 MiB)

```
import java.util.Scanner;
public class e4717 {
        public static void main(String[] args) {
                Scanner in = new Scanner(System.in);
                int n = in.nextInt();
                int k = in.nextInt();
                System.out.println(k % n);
        }
}
```

3.1.3. e0519 Сума квадратів

Знайти суму квадратів двох чисел.

Вхідні дані

Два цілих числа a та b. Числа не перевищують 10^9 за абсолютною величиною.

Вихідні дані Виведіть одне ціле число $a^2 + b^2$.

Складність: 13% — 22162/6902/6104/5307.

На **C++** (5 ms, 1.76 MiB)

```
#include <iostream>
using namespace std;
int main()
{
```

```
        long long a, b;
        cin >> a >> b;
        cout << a*a + b*b;
}
```

На **Python** (22 ms, 5.1 MiB)

```
a,b = map(int,input().split())
print(a**2+b**2)
```

3.1.4. e8877 Повний квадрат

Задано натуральне число **n**. Якщо число **n** є повним квадратом деякого натурального **m**, то виведіть число **m**. У протилежному випадку вивести відповідь **No**.

Вхідні дані Одне натуральне число **n**.

Вихідні дані Виведіть шукану відповідь.

Автор Матвійчук Сергій Володимирович

Джерело Серія задач "Абетка програмування"

Складність: 12% — 3035/1075/1022/901.

Знайдемо цілу частину кореня квадратного з заданого числа. Порівняємо число та квадрат знайденого кореня.

На **Python** (22 ms, 5.1 MiB)

```
n = int(input())
print(m) if n == int(n**.5)**2 else print('No')
```

3.1.5. e0949 Двозначне з чотиризначного

З даного чотиризначного натурального числа створити двозначне, що складається з його середніх цифр.

Вхідні дані

У єдиному рядку задане чотиризначне натуральне число.

Вихідні дані Утворене число.

Складність: 11% — 14956/6694/6090/5425.

На **Java** (298 ms, 24 MiB)

```
import java.util.Scanner;
public class e0949 {
```

```
        public static void main(String[] args) {
                Scanner in = new Scanner(System.in);
                int n = in.nextInt();
                System.out.println(n/10%100);
        }
}
```

На **Python** (22 ms, 5 MiB)

```
print(int(input()) //10 %100)
```

Отримані в програмах числа можуть бути одноцифровими, інакше найкоротшим розв'язком було би **print(input()[1:3])**.

3.1.6. e1213 Масивні числа

Число будемо називати масивним, якщо воно записано у вигляді $a\hat{}n$, що означає піднесення числа a до степеня n. Вам слід порівняти два масивні числа a^b та c^d, записаних у вигляді "основа^експонента".

Із двох заданих масивних чисел слід надрукувати більше.

Вхідні дані

Два масивні числа a та b у вигляді "основа^експонента" (1≤основа, експонента≤1000). Відомо, що значення чисел a та b різні.

Вихідні дані Вивести більше число серед a та b.

Складність: 14% — 4585/1681/1615/1381.

Надамо числа в вигляді "e в степені"(можна "10 в степені") $x^y = e^{y \ln x}$ та порівняємо показники експонент. В умові легка плутанина a та b, водночас розуміємо їх як "масивні"числа.

Python

```
from math import log
a,b = input().split()
ba,ea = map(int,a.split('^'))
bb,eb = map(int,b.split('^'))
print(a) if ea*log(ba)>eb*log(bb) else print(b)
```

3.1.7. e5868 A xor B

Свята простота! Ян Гус

Дано два натуральних числа a та b. Застосуйте до них операцію *побітового виключного або*.

Вхідні дані Два натуральних числа a та b ($a, b \leq 10^9$).

Вихідні дані

Виведіть результат застосування операції *xor* над заданими числами.

Автор Олег Петров

Джерело Літня школа Севастополь 2013, Хвиля 1, День 3

Складність: 4% — 1831/1325/1220/1177.

Програма на **Python**

```
a,b=map(int,input().split())
print(a^b)
```

3.2. Однорядкові програми

Розв'язок в один рядок нагадує гарні математичні формули. Не кожна мова програмування дозволяє таке зробити. Частіше за все необхідною умовою є використання функції введення, а не оператора, процедури чи підпрограми введення. Мовою, що підходить для однорядкових програм є Python. Багато з наведених тут програм не вимагає пояснень.

Тільки виведення

3.2.1. e5232 Метод лінійного перетворення

Ватсону доручили проводити діагностику серцево-судинних захворювань. Для цього йому необхідно опрацьовувати ЕКГ пацієнтів. Він використовує метод лінійного перетворення для сигналу. Форма сигналу може бути описана формулою:

$$F(x) = \int_{-\infty}^{\infty} K(x - x')\Psi(x')dx',$$

де $\Psi(x')$ — функція, що описує положення та інтенсивність окремих компонентів сигналу, $K(x) = \frac{A}{\sigma}e^{-\frac{(x-\mu)^2}{2\sigma^2}}$ — функція Гауса, що описує форму та ширину спектральних ліній. Коефіцієнти μ та σ подаються Ватсону на вхід, а коефіцієнт A тут і надалі розраховується за формулою $A = \frac{1}{\sqrt{2\pi}}$.

Визначення функції Ψ(x) можна подати в вигляді:

$$\Psi(x) = A \int_{-\infty}^{\infty} \frac{\widetilde{F}(y)}{\widetilde{K}(y)} \exp(ixy)dy,$$

де перетворення Фур'є функції $F(x)$ обчислюється як:

$$\widetilde{F}(y) = A \int_{-\infty}^{\infty} F(x) \exp(-ixy)dx.$$

Для проведення точних розрахунків діагностичних задач, Ватсону необхідно визначити числове значення коефіцієнта A.

Вхідні дані

На вхід подається два цілих додатних числа μ та σ, кожне з яких не перевищує 1000.

Вихідні дані

Вивести значення коефіцієнта A з точністю до 4 знаків після десяткової крапки.

Складність цієї задачі: 2% — 520/320/263/259.

Не зважаючи на громіздку умову, уважний її аналіз дозволяє зрозуміти, що відповідь завжди однакова — просто потрібно вивести число 0,3989. Читати дані немає необхідності.

Програма на **Python** (54 ms, 7,88 MiB)

```
print(.3989)
```

Програма на **C++** (2 ms, 1,8 MiB)

```
#include <iostream>
using namespace std;
int main() {
cout << .3989 << endl;
}
```

Програма на **PHP**

```
<?php
  echo .3989;
```

3.2.2. e1024 Hello World!

Виведіть повідомлення "Hello World!".

Python

```
print('Hello World!')
```

Однорядкові задачі

3.3. Введення-виведення

В задачах такого типу немає, або практично немає переробки введених величин.

3.3.1. a0108 Неглухий телефон

Можливо ви колись грали в гру «Глухий телефон», або чули про неї. У цій грі учасникам належало передавати інформацію один одному різним способам: словесно, образно, навіть приходилось писати лівою рукою текст, який інший учасник команди повинен буде прочитати. Також відомо, що практично ніколи інформація не надходить до конечного адресата. Позначимо за $F_i(x)$ функцію, що перетворює текст передавальної інформації x в ту, який отримує учасник $i+1$ від учасника i. Тоді останній n-й учасник отримав дані y, які будуть надаватись наступною формулою: $y = F_{n-1}(F_{n-2}(...F_2(F_1(x))))$.

Але Вам необхідно виключити будь-які зовнішні фактори, що можуть спотворити вхідну інформацію і Ви повинні реалізувати програму «неглухий телефон», що може безпомилково доставляти вхідні дані, тобто в нашому випадку функція $F_i(x)$ для всіх i від 1 до $n-1$.

Вхідні дані В єдиному рядку вхідного файлу INPUT.TXT записано натуральне число від 1 до 100.

Вихідні дані У вихідному файлі OUTPUT.TXT потрібно вивести точно те саме число, задане у вхідному файлі.

Складність задачі: 1%, розв'язуваність 97% (69472)

Словесний розв'язок є в [8] та знаходиться в специфікації Вихідних даних — необхідно просто прочитати вхідні дані та вивести їх.

Програма на **Python**

```
print(input())
```

Pascal

```
var n : shortint;
begin
  assign(input,'input.txt'); reset(input);
  read(n);
  assign(output,'output.txt'); rewrite(output);
  write(n);
end.
```

C

```
#include <stdio.h>
int main(){
  FILE *ifile, *ofile;
  int n;
  ifile = fopen("INPUT.TXT", "r");
  fscanf(ifile, "%d", &n);
  ofile = fopen("OUTPUT.TXT", "w");
  fprintf(ofile, "%d", n);
}
```

3.3.2. e4718 Привіт, Гаррі!

Напишіть програму, яка вітає користувача, виводячи слово Hello, ім'я користувача та розділові знаки у наступному вигляді: Hello, Harry

Вхідні дані У єдиному рядку вводиться ім'я користувача.

Вихідні дані У першому рядку виведіть привітання.

Складність: 6% — 4826/2515/2349/2212.

Програма на **C++**

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    string name;
```

```
    cin >> name;
    cout << "Hello, " << name;
}
```

Програма на **Python** (69 ms, 7.9 MiB)

```
print('Hello, '+input())
```

Розв'язки є і в [16], [25].

3.3.3. e1966 Великий плюс

На сайті у таблиці результатів змагань, які проводяться за правилами ACM (Association for Computing Machinery), вірно розв'язана задачка оцінюється плюсом. Але він якийсь маленький. Виведіть великий плюс з зірочок.

Вхідні дані Ціле число n ($1 \le n \le 100$).

Вихідні дані Виведіть відповідний великий квадратний "плюс"з точок та зірочок – див. приклади вхідних та вихідних даних.

Вхідні дані #1	Вихідні дані #1	Вхідні дані #2	Вихідні дані #2
1	.*. *** .*.	2	..*.. ..*.. ***** ..*.. ..*..

Джерело II етап Всеукраїнської олімпіади школярів 2011-2012, м. Бердичів.

Складність: 6% — 2535/1537/1327/1250.

Python (40 ms, 5,5 MiB)

Скористаємось множенням рядка. В циклі виведемо рядки вертикальної частини великого плюсу. Виводимо горизонтальний рядок. Потім знову виведемо рядки вертикальної частини великого плюсу. Навряд чи тут є сенс робити функцію/процедуру для вертикальної частини плюсу.

```
n=int(input())
for i in range(n): print('.'*n+'*'+'.'*n)
print('*'*(2*n+1))
for i in range(n): print('.'*n+'*'+'.'*n)
```

Розв'язок є і в [16].

3.3.4. e1119 Піраміда з символів

Вася хоче надрукувати на принтері піраміду з якогось символу висотою h. Напишіть програму, яка допоможе йому у цьому, не забуваючи, що програма повинна бути "економічно вигідною тобто друкувати найменшу кількість символів.

Приклади пірамід наведено у прикладах вхідних та вихідних даних.

Вхідні дані #1
```
A 3
```
Вихідні дані #1
```
12
  A
 AAA
AAAAA
```
Вхідні дані #2
```
M 9
```
Вихідні дані #2
```
117
        M
       MMM
      MMMMM
     MMMMMMM
    MMMMMMMMM
   MMMMMMMMMMM
  MMMMMMMMMMMMM
 MMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMM
```

Джерело II етап Всеукраїнської олімпіади 2010-2011 м. Бердичів

Складність: 11% — 4816/1962/1718/1530.

Кількість символів в кожному рядку піраміди (враховуючи пробіли на початку рядка): $h, h+1, h+2, ...2h-1$ — представляє собою арифметичну прогресію (2.1.3). Її сума $\frac{(h+(2h-1))}{2}h$.

Python

```
s,h=input().split()
h=int(h)
print(h*(3*h-1)//2)
for i in range(h): print('␣'*(h-i-1)+s*(2*i+1))
```

Pascal

```
var h,i,j:longint;
    s:char;
begin
    readln(s,h);
```

```
    writeln((h*(3*h-1)) div 2);
    for i:=1 to h do
        begin
            for j:=1 to h-i do write('␣');
            for j:=1 to 2*i-1 do write(s);
            writeln;
        end;
end.
```

C#

```
using System;
class Program{
  static void Main(string[] args){
    string input = Console.ReadLine();
    var inp = input.Split('␣');
    string a = inp[0];
    int h = Convert.ToInt32(inp[1]);
    int count = 0, tempcount;
    string temp = string.Empty;
    tempcount = (h - 1);
    int let = 1;
    while (tempcount >= 0){
      for(int i = 0; i < tempcount; i++){
         temp += '␣';
         count++;
      }
      for(int i = 0; i < let; i++){
         temp += a;
         count++;
      }
      temp += '\n';
      tempcount--;
      let += 2;
    }
    temp = temp.Substring(0, temp.Length - 1);
    string res = count.ToString() + "\n" + temp + "\n";
    Console.Write(res);
  }
}
```

3.3.5. e9404 Квадрати та діагоналі

Дивіться приклад ...

Вхідні дані Одне ціле число n ($3 \leq n \leq 49$).

Input #1	**Output #1**
3	*** *** ***

Input #2	**Output #2**
7	******* ** ** * * * * * * * * * * * ** ** *******

Складність: 0% — 98/79/54/54.

На **Python** (20 ms, 5.1 MiB)

```
n = int(input())
print('*'*n)
for i in range((n-3)//2):
    print('*'+' '*i+'*'+' '*(n-4-2*i)+'*'+' '*i+'*')
print('*'+' '*((n-3)//2)+'*'+'*'*((n+1)%2)+' '*((n-3)\
  //2)+'*')
for i in range((n-4)//2,-1,-1):
    print('*'+' '*i+'*'+' '*(n-4-2*i)+'*'+' '*i+'*')
print('*'*n)
```

3.3.6. e1340 Алмаз

Намалюйте алмаз ромбовидної форми при допомозі символів '*'.

Вхідні дані

У вхідних даних міститься декілька рядків тестових даних. Кожен тест у окремому рядку містить єдине ціле невід'ємне число N ($N \leq 100$), рядок зі значенням N рівним нулю сигналізує про завершення вхідних даних.

Вихідні дані Для кожного випадку вхідних даних надрукуйте зразок алмазу шириною N як у прикладі. Звертаємо увагу, що у кожному рядку не повинно бути зайвих пропусків після символів '*'.

Input	Output
3	* *** ***** *** *
2	* *** *
0	

Складність: 21% — 4287/1074/1146/903.

Програма на **Python** (51 ms, 5.5 MiB)

```
f=0
while 1:
    n=int(input())
    if(not n): break
    if f: print()
    for i in range(n):
        print('␣'*(n-i-1),'*'*(2*i+1),sep='')
    for i in range(1,n):
        print('␣'*i,'*'*(2*(n-i)-1),sep='')
    f=1
```

3.3.7. e8938 #Прямокутник

Задано натуральне число n. Вивести прямокутник розміром $n \times 3$ з символів # як показано у прикладі.

Вхідні дані Одне натуральне число n ($n \leq 100$).

Вихідні дані Вивести прямокутник розміром $n \times 3$ із символів #.

Input	Output
2	### ###

Автор Матвійчук Сергій Володимирович

Джерело Серія задач "Абетка програмування"

Складність: 4% — 1856/1291/1114/1074.

Програма на **Python** (18 ms, 5.1 MiB)

```
for i in range(int(input())): print('###'))
```

3.3.8. e8939 #Прямокутник 2

Задано натуральне число n. Вивести прямокутник розміром $4 \times n$ із символів #, як показано у прикладі.

Вхідні дані Одне натуральне число n ($n \leq 100$).

Вихідні дані Вивести прямокутник розміром $4 \times n$ із символів #.

Input	**Output**
2	## ## ## ##

Автор Матвійчук Сергій Володимирович

Джерело Серія задач "Абетка програмування"

Складність: 6% — 1782/1063/895/837.

Програма на **Python** (20 ms, 5.1 MiB)

```
n = int(input())
for i in range(4): print('#'*n)
```

3.3.9. e8940 #Прямокутник 3

Задано два натуральних числа n та m. Вивести прямокутник розміром $n \times m$ із символів #, як показано у прикладі.

Вхідні дані Два натуральних числа n та m ($n, m \leq 100$).

Вихідні дані Вивести прямокутник розміром $n \times m$ із символів #.

Input	**Output**
2 3	### ###

Автор Матвійчук Сергій Володимирович

Джерело Серія задач "Абетка програмування"

Складність: 5% — 1217/849/722/683.

Програма на **Python** (23 ms, 5.1 MiB)

```
n,m = map(int,input().split())
for i in range(n): print('#'*m)
```

3.3.10. e8942 *Рамка

Для заданого натурального числа n вивести горизонтальну прямокутну рамку розміром $3 \times n$ із зірочок, заповнену проміжком (як показано у прикладі).

Вхідні дані Одне натуральне число n ($n \leq 100$).

Вихідні дані Виведіть прямокутну рамку розміром $3 \times n$.

Input	**Output**
5	***** * * *****

Автор Матвійчук Сергій Володимирович

Джерело Серія задач "Абетка програмування"

Складність: 6% — 1404/788/689/649.

Програма на **Python** (25 ms, 5.1 MiB)

```
n = int(input())
print('*'*n)
print('*'+' '*(n-2)+'*'*(n>1))
print('*'*n)
```

3.3.11. e8943 *Рамка 2

Для заданого натурального числа n вивести горизонтальну прямокутну рамку розміром $n \times 3$ із зірочок, заповнену проміжком як показано у прикладі.

Вхідні дані Одне натуральне число n ($n \leq 100$).

Вихідні дані Виведіть прямокутну рамку розміром $n \times 3$.

Input	**Output**
5	*** * * * * * * ***

Автор Матвійчук Сергій Володимирович

Джерело Серія задач "Абетка програмування"

Складність: 10% — 1420/660/621/556.

Програма на **Python** (20 ms, 5.1 MiB)

```
n = int(input())
print('***')
for i in range(n-2): print('* *')
```

```
if n>1: print('***')
```

3.3.12. e8944 *Рамка 3

Для заданого натурального числа n вивести квадратну рамку розміром $n \times n$ із зірочок, заповнену проміжком як показано у прикладі.

Вхідні дані Одне натуральне число n ($n \leq 100$).

Вихідні дані Виведіть прямокутну рамку розміром $n \times n$.

Input	Output
5	***** * * * * * * *****

Автор Матвійчук Сергій Володимирович

Джерело Серія задач "Абетка програмування"

Складність: % — 1321/625/563/497.

Програма на **Python** (23 ms, 5.1 MiB)

```
n = int(input())
print('*'*n)
for i in range(n-2): print('*'+' '*(n-2)+'*'*(n>1))
if n>1: print('*'*n)
```

3.3.13. e8945 *Рамка 4

Для заданих натуральних чисел n та m вивести прямокутну рамку розміром $n \times m$ із зірочок, заповнену проміжком як показано у прикладі.

Вхідні дані Два натуральних числа n і m ($n, m \leq 100$).

Вихідні дані Виведіть прямокутну рамку розміром $n \times m$.

Input	Output
4 7	******* * * * * *******

Автор Матвійчук Сергій Володимирович

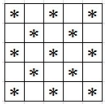
Джерело Серія задач "Абетка програмування"

Складність: 12% — 1321/625/563/497.

Програма на **Python** (19 ms, 5.1 MiB)

```
n,m = map(int,input().split())
print('*'*m)
for i in range(n-2): print('*'+' '*(m-2)+'*'*(m>1))
if n>1: print('*'*m)
```

3.3.14. e8946 Шаблон



За заданим натуральним числом n вивести зображення розміром $n \times n$, утворене символами зірочка та проміжок як показано у прикладі.

Вхідні дані Одне натуральне число n.

Вихідні дані Вивести зображення $n \times n$.

Автор Матвійчук Сергій Володимирович

Джерело Серія задач "Абетка програмування"

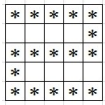
Складність: 6% — 1348/798/655/613.

Будемо "замощувати"блоками (пробіл, зірочка), і зірочка на початку, якщо потрібна.

Програма на **Python** (17 ms, 5.1 MiB)

```
n = int(input())
for i in range(n):
    print('*'*((i+1)%2)+' *'*((n-1+i%2)//2))
```

3.3.15. e8947 Шаблон 2



За заданим натуральним числом n вивести зображення розміром $n * n$, утворене символами зірочка та проміжок як показано у прикладі.

Вхідні дані Одне натуральне число n.

Вихідні дані Вивести зображення $n \times n$.

Автор Матвійчук Сергій Володимирович

Джерело Серія задач "Абетка програмування"

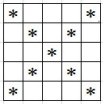
Складність: 10% — 1080/629/550/494.

За шаблоном не зовсім зрозуміло його продовження. З'ясувалось що це "змійка".

Програма на **Python** (21 ms, 5.1 MiB)

```
n = int(input())
for i in range(1,n+1):
  print('*'*n) if i%2 else print('␣'*(n-1)*(i%4//2)+'*')
```

3.3.16. e8948 Шаблон 3



За заданим натуральним числом n вивести зображення розміром $n \times n$, утворене символами зірочка та проміжок як показано у прикладі.

Вхідні дані Одне натуральне число n ($n > 1$).

Вихідні дані Вивести зображення $n \times n$.

Автор Матвійчук Сергій Володимирович

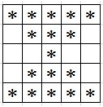
Джерело Серія задач "Абетка програмування"

Складність: 4% — 803/590/490/468.

Програма на **Python** (50 ms, 5.1 MiB)

```
n = int(input())
for i in range(n//2):
    print('␣'*i+'*'+'␣'*(n-2-2*i)+'*')
if n%2: print('␣'*(n//2)+'*')
for i in range(n//2-1,-1,-1):
    print('␣'*i+'*'+'␣'*(n-2-2*i)+'*')
```

3.3.17. e8949 Шаблон 4



За заданим непарним натуральним числом n вивести зображення розміром $n \times n$, утворене символами зірочка та проміжок як показано у прикладі.

Вхідні дані Одне непарне натуральне число n ($n > 1$).

Вихідні дані Вивести зображення $n \times n$.

Автор Матвійчук Сергій Володимирович

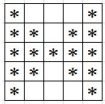
Джерело Серія задач "Абетка програмування"

Складність: 9% — 792/463/395/361.

Програма на **Python** (28 ms, 5.1 MiB)

```
n = int(input())
for i in range(n//2+1): print('␣'*i+'*'*(n-2*i))
for i in range(n//2): print('␣'*(n//2-i-1)+'*'*(2*i+3))
```

3.3.18. e8950 Шаблон 5



За заданим непарним натуральним числом n вивести зображення розміром $n \times n$, утворене символами зірочка та проміжок як показано у прикладі.

Вхідні дані Одне непарне натуральне число n ($n > 1$).

Вихідні дані Вивести зображення $n \times n$.

Автор Матвійчук Сергій Володимирович

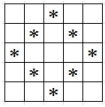
Джерело Серія задач "Абетка програмування"

Складність: 5% — 609/452/363/345.

Програма на **Python** (32 ms, 5.1 MiB)

```
n = int(input())
for i in range(n//2+1): print('*'*i+'␣'*(n-2*i)+'*'*i)
print('*'*n)
for i in range(n//2):
    print('*'*(n//2-i)+'␣'*(2*i+1)+'*'*(n//2-i))
```

3.3.19. e8951 Шаблон 6



За заданим непарним натуральним числом n вивести зображення розміром $n*n$, утворене символами зірочка та проміжок як показано у прикладі.

Вхідні дані Одне непарне натуральне число n ($n > 1$).

Вихідні дані Вивести зображення $n \times n$.

Автор Матвійчук Сергій Володимирович

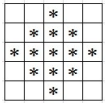
Джерело Серія задач "Абетка програмування"

Складність: 6% — 458/316/309/290.

Програма на **Python** (22 ms, 5.1 MiB)

```
n = int(input())
for i in range(n//2+1):
    print('␣'*(n//2-i)+'*'+'␣'*(2*i-1)+'*'*(i>0))
for i in range(n//2-1,-1,-1):
    print('␣'*(n//2-i)+'*'+'␣'*(2*i-1)+'*'*(i>0))
```

3.3.20. e8952 Шаблон 7



За заданим непарним натуральним числом n вивести зображення розміром $n*n$, утворене символами зірочка та проміжок як показано у прикладі.

Вхідні дані Одне непарне натуральне число n ($n > 1$).

Вихідні дані Вивести зображення $n \times n$.

Автор Матвійчук Сергій Володимирович

Джерело Серія задач "Абетка програмування"

Складність: 6% — 458/316/309/290.

Використовуємо, спираємось на попередню 3.3.19

Програма на **Python** (18 ms, 5.1 MiB)

```
n = int(input())
for i in range(n//2+1):
    print(' '*(n//2-i)+'*'*(2*i+1))
for i in range(n//2,-1,-1):
    print(' '*(n//2-i+1)+'*'*(2*i-1))
```

3.3.21. e1124 Алфавітне графіті

Графіті — один з видів сучасного варварського живопису. Вася, як і належить гідному нащадку варварів, вирішив також зайнятись цією досить перспективною з його точки зору справою і увіковічити своє перебування у школі написами у стилі графіті.

Так як по малюванню у Васі була тверда "двійка а він почав ще й вивчати англійську, після вивчення досить важко піддавшийся йому у вимові третьої літери англійського алфавіту він вирішив увіковічити свої муки у вивченні цього знову ж таки важкого для нього предмету відображенням кожного етапу, пов'язаного з вивченням чергової літери, відповідним написом на шкільній стіні.

Художні митарства Васі після вивчення 3-ї літери наведено у прикладі вихідних даних.

Так як з кожним етапом зробити вірний напис Васі ставало все важче і важче, напишіть програму, яка допоможе йому зробити шпаргалку для нанесення чергового візерунку.

Вхідні дані Єдине число N ($3 \le N \le 26$)) — кількість вивчених літер англійського алфавіту.

Вихідні дані Напис на стіні, зроблений Васею після вивчення N літер англійського алфавіту.

Input **Output**

3	a a a ab aabc

Складність: 7% — 4424/2054/1793/1674.

Програма на **Python** (22 ms, 5.1 MiB)

```
n,a=int(input()),'abcdefghijklmnopqrstuvwxyz'
for i in range(n): print('a','␣'*(n-i-1),a[:i+1],sep='')
```

Теорія чисел

3.4. Цілі числа

3.4.1. e1286 Шкільний буфет

У шкільному буфеті до завершення уроків залишилось декілька тістечок: **a** ванільних, **b** шоколадних і **c** фруктових. Дмитро збирається придбати тістечка перед закриттям буфету. Яку найбільшу кількість тістечок може вибрати Дмитро?

Вхідні дані В одному рядку задано три цілих невід'ємних числа — відповідні кількості тістечок, кожне з яких не перевищує 20000.

Вихідні дані Вивести найбільшу кількість тістечок, яку може вибрати Дмитро.

Складність: 4% — 7558/5129/4584/4394.

Найбільша кількість — все, що є в буфеті.

Python

```
print(sum(map(int,input().split())))
```

Код програми збігається з кодами 3.4.11,

3.4.2. e6277 Покупка води

Вартість пляшки води, враховуючи вартість порожньої пляшки, становить **1 грн 20 коп.**, а вартість порожньої пляшки **20 коп**.

Скільки пляшок води можна випити на n грн, враховуючи, що порожні пляшки можна здавати, і на одержані гроші купувати нові пляшки води.

Вхідні дані Натуральне значення n ($1 \le n \le 1000$).

Вихідні дані Кількість пляшок води, яку можна випити на n грн.
Складність: 10% — 10589/4852/4085/3672.

Вода в одній пляшці коштує 1 грн, отже можливо було би випити n літрів. Водночас на купівлю першої пляшки потрібно на гривню більше, таким чином всього можна купити/випити $n-1$ л води.

Python (22 ms, 5.5 MiB)

```
print(int(input())-1)
```

Розв'язок є і в [16].

3.4.3. e9539 Задача про Вову

Висота Вови на каблуках і в капелюсі **a** см, на каблуках без капелюха — **b** см, а в капелюсі без каблуків — **c** см. Який реальний зріст Вови?
Вхідні дані В єдиному рядку записані через пропуск три натуральних числа **a**, **b** і **c**. Числові значення коректні і не перевищують 200.
Вихідні дані Одне число — реальний зріст Вови.
Складність: 8% — 1075/714/674/620.

Складаємо систему рівнянь і знаходимо **h=b+c-a**.

На **Python** (24 ms, 5.1 MiB)

```
a,b,c = map(int,input().split())
print(b+c-a)
```

3.4.4. e5175 Остання цифра

Вивести останню цифру числа.
Вхідні дані Одне натуральне число n ($1 \le n \le 10^9$).
Вихідні дані Виведіть останню цифру числа n.
Складність задачі: 3% — 8105/5280/4173/4036.

Як ми знаємо (2.1.1), для натуральних чисел () остання цифра є залишком ділення числа на 10.

На **C++** (3 ms, 1.75 MiB)

```
#include <iostream>
using namespace std;
int main(){
        long n;
```

```
        cin >> n;
        cout << n % 10;
}
```

На **Python** (25 ms, 7.7 MiB)

```
print(int(input())%10)
```

3.4.5. e0001 Проста задача

Програма зчитує двоцифрове число і виводить через пропуск кожну цифру окремо.

Вхідні дані

Натуральне число на проміжку від 10 до 99 включно.

Вихідні дані

Спочатку першу цифру числа і через пробіл другу.

Складність задачі — 11% — 127394/50709/40443/36058.

Використовуємо, спираємось на **??**.

Оскільки число натуральне, перша його цифра — ділення числа націло на 10, друга цифра (остання) — залишок ділення на 10.

На **C**

```
#include<stdio.h>
int main() {
    int n;
    scanf("%d", &n);
    printf("%d␣%d\n", n/10, n%10);
}
```

Аналогічно в [31].

На **C++** (1 ms, 1.6 MiB)

```
#include <iostream>
using namespace std;
int main() {
        int n;
        cin >> n;
        cout << n/10 << "␣" << n%10 << endl;
}
```

Python

```
n=int(input())
print(n//10,n%10)
```

Схожий розв'язок в [31]. Інший розв'язок — введемо число як текст, перетворимо в список символів (їх буде 2), об'єднаємо їх через пробіл, виведемо результат:

```
print('␣'.join(list(input())))
```

Pascal [31]

```
var n:integer;
begin
  read(n);
  writeln(n div 10,'␣',n mod 10)
end.
```

C# [31] (90 ms, 7 MiB)

```
using System;
class Solution {
  public static void Main(string[] args) {
    int n = Convert.ToInt32(Console.ReadLine());
    Console.WriteLine("{0}␣{1}", (int)n/10, (int)n%10);
  }
}
```

Java [31]

```
import java.util.Scanner;
public class Main {
  public static void main(String[] args) {
    Scanner in = new Scanner(System.in);
    int number = in.nextInt();
    System.out.printf("%d␣%d\n", number/10, number%10);
  }
}
```

Аналогічно маємо (125 ms, 24 MiB)

```
import java.util.Scanner;
public class e0001 {
  public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int n = in.nextInt();
        System.out.println(n/10 +" " + n%10);
  }
}
```

PHP (42 ms, 13 MiB)

```
<?php
  $n = intval(trim(fgets(STDIN)));
  echo sprintf("%d %d\n", $n/10, $n%10);
```

Також в [31]. Там же є і інший варіант — обробляти число як текст

```
<?php
 $input = trim(file_get_contents('php://stdin'));
 echo $input{0}.' '.$input{1}.PHP_EOL;
```

```
<?php
 $input = trim(stream_get_contents(STDIN));
 echo $input{0}.' '.$input{1}.PHP_EOL;
```

JavaScript [31]

```
process.stdin.on('data', function(data){
  let n = parseInt(data, 10)
  console.log(`${Math.floor(n/10)} ${n%10}`);
});
```

Ruby [31]

```
number = gets.chomp
a = (number.to_i / 10)
b = (number.to_i % 10)
puts a.to_s + " " + b.to_s
```

Go [31]

```
package main
import "fmt"
func main() {
  var v int
  fmt.Scanf("%d", &v)
  fmt.Printf("%d %d\n", v / 10, v % 10)
}
```

Haskell [31]

```
main = do
    input <- readLn :: IO Int
    let a = input `div` 10
    let b = input `mod` 10
    putStrLn $ (show a) ++ " " ++ (show b)
```

3.4.6. e1606 Сума першої та останньої цифр числа

Знайти суму першої та останньої цифри цілого числа.

Вхідні дані Одне ціле 32-х розрядне число, що містить не менше 2-х цифр.

Вихідні дані Одне число - розв'язок задачі.

Складність: 13% — 9837/3888/3300/2882.

Використовуємо, спираємось на 3.4.5, 3.10.2. Не забуваємо, що ціле число може бути від'ємним. Вводимо число, переводимо в додатнє, знову переводимо в рядок, виділяємо перший та останній символ-цифру, переводимо їх в цілі числа, підсумовуємо, виводимо.

Програма на **Python**

```
n=str(abs(int(input())))
print(int(n[0])+int(n[len(n)-1]))
```

Альтернативно вводимо число, переводимо в додатнє, виділяємо останню цифру, все число переводимо в рядок і беремо перший символ, переводимо в ціле число. Підсумовуємо два отриманих числа, підсумовуємо, виводимо.

```
n=abs(int(input()))
print(int(str(n)[0])+n%10)
```

3.4.7. e0002 Цифри

Підрахувати кількість цифр цілого невід'ємного числа n.

Вхідні дані Одне ціле невід'ємне число n ($0 \leq n \leq 2 \cdot 10^9$).

Вихідні дані Виведіть кількість цифр у числі n.

Складність: 24% — 102389/25983/26587/20309.

Використовуємо, спираємось на **??**, **??**. Алфавіт числа в десятковій системі числення —– 0,1,...,9. Простий варіант, але не самий короткий за кодом: в циклі перераховуємо потрібні символи.

Програма на **Pascal**

```
var n:string;
    k,i:byte;
begin
      readln(n);
      for i:=1 to length(n) do
          if n[i] in ['0'..'9'] then inc(k);
      writeln(k);
end.
```

Щоб позбути зайвих символів переведемо введене в ціле число, потім назад в рядок і порахуємо його довжину, що і є кількістю цифр. Програма на **Python** (42 ms, 5,1 MiB)

```
print(len(str(int(input()))))
```

Розв'язок є і в [16].

Трішечки модифікований розв'язок на C++ з [16]

```
#include <bits/stdc++.h>
using namespace std;
int main() {
        int n,k=0;
        cin >> n;
        do{n /= 10; k++;} while(n);
        cout << k << endl;
}
```

3.4.8. e0903 Перша чи остання?

Задано трицифрове число. Визначити, яка цифра в ньому є більшою – перша чи остання.

Вхідні дані У єдиному рядку задано трицифрове число.

Вихідні дані

Вивести більшу з вказаних цифр. У випадку їх рівності вивести знак "=" (без лапок).

Складність: 8% — 32841/14113/11952/10960.

Pascal

```
var n,r:string;
begin
  read(n);
  if n[1]>n[3] then r:=n[1] else
  if n[1]<n[3] then r:=n[3] else r:='=';
  writeln(r)
end.
```

3.4.9. e0906 Добуток цифр

Задано трицифрове число. Визначити добуток його цифр.

Вхідні дані

Одне додатнє трицифрове число n.

Вихідні дані

Вивести добуток цифр числа n.

Джерело ДПА-2010 Варіант 6

Вхідні дані #1 235 **Вихідні дані #1** 30

Складність: 5% — 29401/16694/13451/12828.

Pascal

```
var n,i:integer;
begin
      read(n);
      i:=n div 100;
      writeln(i*((n-i*100)div 10)*(n mod 10));
end.
```

C/C++ (2 ms, 1,75 MiB)

```
#include <iostream>
using namespace std;
int main()
{
        int n;
        cin >> n;
        cout << (n / 100)*(n % 100 / 10)*(n % 10);
}
```

3.4.10. e0961 Чотирицифрове без середніх

Записати задане чотирицифрове натуральне число без середніх цифр.

Вхідні дані

У єдиному рядку записано задане натуральне чотирицифрове число.

Вихідні дані Очікувана відповідь.

Вхідні дані #1 1023 **Вихідні дані #1** 13

Складність: 2% — 4148/3119/2880/2824.

Python

```
s=input()
print(s[0],s[3],sep='')
```

3.4.11. a0001 c0100 A + B

Потрібно скласти два цілих числа A и B.

Вхідні дані

В єдиному рядку вхідного файлу INPUT.TXT записані два натуральних числа через пробіл. Значення чисел не перевищують 10^9.

Вихідні дані

В єдиному рядку вихідного файлу OUTPUT.TXT потрібно вивести одне ціле число — суму чисел A і B.

Складність задачі: 2%, розв'язуваність 91% (110004).

Pascal (0.064 с, 964 Кб)

```
var a,b:longint;
begin
  read(a,b);
  write(a+b);
end.
```

C (0.011 с, 60 Кб)

```c
#include <stdio.h>
long a,b;
int main(){
  freopen("input.txt","r",stdin);
  freopen("output.txt","w",stdout);
  scanf("%ld%ld",&a,&b);
  printf("%ld",a+b);
}
```

Програма на **Python**

```python
print(sum(map(int,input().split())))
```

3.4.12. e1001 A + B у двійковій системі числення

Обчисліть **A** + **B** у двійковій системі числення.

Вхідні дані

В окремих рядках у двійковій системі числення задано два невід'ємних цілих числа **A** та **B**. Кількість цифр у кожному з чисел не перевищує 1000.

Вихідні дані

Виведіть суму **A** + **B** у двійковій системі числення.

Оскільки підрозуміється довга арифметика скористуємось Python. Введемо кожний рядок як двійкове число (int(input(),2)), підсумуємо їх, перетворимо у двійкове число (bin()) та виведемо без початкових символів (префіксу), що позначають двійкове подання.

Програма на **Python**

```python
print(bin(int(input(),2)+int(input(),2))[2:])
```

3.4.13. e1603 Сума цифр числа

Знайти суму цифр цілого числа.

Вхідні дані Одне ціле 32-х розрядне число n (число може бути від'ємним).

Вихідні дані Вивести суму цифр числа n.

Складність: 14% — 13355/4820/4111/3526.

Вводимо число, переводимо в додатнє, знову переводимо в рядок і переводимо в список символів (цифр) (list()). Надалі переводимо їх в числа (map(int,)) і нарешті підсумовуємо їх.

Програма на **Python**

```
print(sum(map(int,list(str(abs(int(input()))))))))
```

Деякий розв'язок є в [16].

3.4.14. e0933 Сума цифр двоцифрового числа

Знайти суму цифр даного двоцифрового числа. **Вхідні дані** У єдиному рядку задане двоцифрове ціле число.

Вихідні дані У єдиному рядку сума його цифр.

Складність: 18% — 28302/7610/7734/6306.

Pascal

```
var n:integer;
begin
  read(n);
  writeln(abs(n div 10)+abs(n mod 10))
end.
```

Java (188 ms, 25.1 MiB)

```
import java.util.Scanner;
public class e0933 {
        public static void main(String[] args) {
                Scanner in = new Scanner(System.in);
                int n = Math.abs(in.nextInt());
                System.out.println(n/10+n%10);
        }
}
```

Використовуємо, спираємось на 3.4.13 **Python**

```
print(sum(map(int,list(str(abs(int(input()))))))))
```

Деякий розв'язок є в [16].

3.4.15. e2804 Квадратні числа

Назвемо число *квадратним*, якщо сума його цифр є квадратом деякого натурального числа.

Наприклад, число 88 є квадратним, так як сума його цифр 8+8=16, а це квадрат числа 4, а число 23 таким не буде, так як сума його цифр 2+3=5, і не існує натурального числа, квадрат якого дорівнює 5.

Перевірте, чи є задане натуральне число квадратним.

Вхідні дані Єдине натуральне число, яке не перевищує 10^9.

Вихідні дані Виведіть **Yes**, якщо задане число є квадратним, і **No** у протилежному випадку.

Автор Анатолій Присяжнюк

Джерело II етап Всеукраїнської олімпіади школярів 2012-2013, м. Бердичів

Складність: 15% — 2032/731/777/657.

Використовуємо, спираємось на 2.2, 3.4.13.

Зауважимо, що квадратні числа даної задачі відрізняються від загальноприйнятих (2.1.1).

Знаходимо суму цифр. Далі порівнюємо квадрат цілої частини кореня квадратного суми з самою сумою.

Програма на **Python** (43 ms, 5,5 MiB)

```
s=sum(map(int,list(input())))
print('Yes') if int(s**.5)**2==s else print('No')
```

3.4.16. e8888 Наступне парне число

Дано ціле число n. Вивести наступне парне до числа n.

Вхідні дані Одне ціле число n.

Вихідні дані Вивести парне число після n.

Автор Матвійчук Сергій Володимирович

Джерело Серія задач "Абетка програмування"

Складність: 4% — 935/685/654/628.

Збільшимо вхідне число на 2 (потрібно наступне), поділимо націло на 2 (при цьому відкидається дробова частина, коли число непарне) та помножимо на 2 (отримуємо парне).

Програма на **Python** (19 ms, 5,1 MiB)

```
print((int(input())+2)//2*2)
```

3.4.17. e8887 Наступне непарне число

Дано ціле число n. Вивести наступне непарне до числа n.
Вхідні дані Одне ціле число n.
Вихідні дані Вивести непарне число після n.
Автор Матвійчук Сергій Володимирович
Джерело Серія задач "Абетка програмування"
Складність: 6% — 847/556/557/526.
Використовуємо, спираємось на 3.4.16.

Програма на **Python** (19 ms, 5,1 MiB)

```
print((int(input())+1)//2*2+1)
```

3.4.18. e8886 Попереднє парне число

Дано ціле число n. Вивести попереднє парне до числа n.
Вхідні дані Одне ціле число n.
Вихідні дані Вивести парне число перед n.
Автор Матвійчук Сергій Володимирович
Джерело Серія задач "Абетка програмування"
Складність: 5% — 1933/1192/1071/1018.

Використовуємо, спираємось на 3.4.16, 3.4.17. Зменшимо вхідне число (потрібно попереднє), поділимо націло на 2 та помножимо на 2 (отримуємо парне).

Програма на **Python** (19 ms, 5,1 MiB)

```
print((int(input())-1)//2*2)
```

3.4.19. e8885 Попереднє непарне число

Дано ціле число n. Вивести попереднє непарне до числа n.
Вхідні дані Одне ціле число n.
Вихідні дані Вивести непарне число перед n.
Автор Матвійчук Сергій Володимирович
Джерело Серія задач "Абетка програмування"
Складність: 7% — 2563/1268/1226/1142.

Використовуємо, спираємось на 3.4.16, 3.4.17, 3.4.18.

Програма на **Python** (20 ms, 5,1 MiB)

```
print((int(input())-2)//2*2+1)
```

3.4.20. e7817 Гарне число

"Гарним"будемо вважати число, що складається лише з непарних цифр. Наприклад число **157953** гарне, а число **2452117** ні. Необхідно з'ясувати, скільки існує n-значних гарних чисел.

Вхідні дані Одне ціле невід'ємне число n ($1 \le n \le 20$).

Вихідні дані Вивести кількість гарних чисел.

Вхідні дані #1 4

Вихідні дані #1 625

Складність: 11% — 2442/1165/1054/940.

В одному розряді числа може бути одне з 5 непарних чисел. Таким чином потрібних n-розрядних чисел може бути 5^n.

Програма на **Python** (30 ms, 7,7 MiB)

```
print(5**int(input()))
```

Розв'язок є і в [16].

3.4.21. e0852 Ділення довгого числа на коротке

Задано ціле невід'ємне число m і ціле додатнє число n. Знайти m div n та m mod n.

Вхідні дані У першому рядку знаходиться число m, у другому n ($0 \le m \le 10^{60000}$, $1 \le n \le 1\,000\,000$).

Вихідні дані

У першому рядку вивести значення виразу $m\ div\ n$, у другому — значення виразу $m\ mod\ n$.

Складність: 19% — 1231/466/483/389.

Виконаємо дії безпосередньо.

Програма на **Python**

```
m,n = int(input()), int(input())
print(m // n)
print(m % n)
```

3.4.22. e1315 A + B

Задано два цілих числа **A** та **B** у незвичному поданні: числа розділені на групи по 3 цифри, починаючи з розряду одиниць, комами. Ваше завдання знайти їх суму і вивести результат у звичній нормальній десятковій формі виведення.

Вхідні дані Вхідні дані складаються з декількох наборів даних. Кожен тестовий приклад подано у окремому рядку, містить 2 цілих числа **A** та **B** ($|\mathbf{A}|$, $|\mathbf{B}| < 10^9$) і числа відокремлені пропуском.

Вихідні дані Для кожного тестового прикладу вивести результат у окремому рядку у нормальній формі виведення.

Input	**Output**
-234,567,890 123,456,789	-111111101
1,234 2,345,678	2346912

Складність: 11% — 537/252/247/219.

Програма на **Python** (145 ms, 8 MiB)

```
while 1:
  try:
    ab=input().replace(',','')
    print(sum(map(int,ab.split())))
  except:
    break
```

3.4.23. e0518 Сума двох

Знайти суму двох чисел.

Вхідні дані

Перший рядок містить кількість тестів t ($1 \leq t \leq 100$). Кожен тест складається з двох цілих чисел a та b.

Вихідні дані

Для кожного тесту вивести в окремому рядку суму чисел a та b.

Input	**Output**
3	5
2 3	-1
17 -18	11
5 6	

Складність: 8% — 8778/4441/3748/3451.

Програма на **Pascal** (2 ms, 0.68 MiB)

```
var t,i,a,b:int32;
begin
  readln(t);
  for i:=1 to t do
    begin
      read(a,b);
      writeln(a+b)
    end
end.
```

3.4.24. e0313 A + B

Петрику задали домашнє завдання: знайти суму 2-х натуральних чисел A та B.

Вхідні дані

У першому рядку задано кількість заданих Петрику прикладів N, а далі йде N рядків у форматі A+B, де A та B — 2 заданих натуральних числа, а між ними без пропусків символ виконання дії додавання "+".

Відповідність вхідних даних вказаному формату гарантується (див. приклад вхідних даних). Вхідні дані не перевищують 10^{500}. ($0 < N \leq 250$)

Вихідні дані

У N рядках вивести шукані суми.

Input	**Output**
2	8
5+3	14818641113332977
14818641113280510+52467	

Складність: 26% — 5324/1692/1890/1394.

Програма на **Python**

```
n=int(input())
for i in range(n):
  a=input().split('+')
  print(int(a[0])+int(a[1]))
```

3.4.25. e0314 A + B ?

Петрику знову задали таке ж саме домашнє завдання: найти суму 2-х натуральних чисел A та B. Але, так як Петрик був завзятим рибалкою і

хотів потрапити на вечірню ловлю, він переписував завдання поспіхом, правда робив завжди одну помилку: інколи замість знаку додавання «+», він ставив знак віднімання «-». Увечері, після рибалки, сівши виконувати домашнє завдання, він почав виконувати всі приклади саме так, як у нього було записано у зошиті.

А ви зможете виконати це ж завдання?

Вхідні дані

У першому рядку задано кількість заданих Петрику прикладів N, а далі йдуть N рядків у форматі A+B, де A та B — 2 заданих натуральних числа, а між ними без пропусків символ виконання дії додавання «+» або віднімання «-».

Відповідність вхідних даних вказаному формату гарантується (див. приклад вхідних даних). Вхідні дані не перевищують 10^{500}. $0 < N \leq 800$.

Вихідні дані В N рядках вивести шукані відповіді для прикладів.

Input	**Output**
2	8
5+3	14818641113228043
14818641113280510-52467	

Складність: 30% — 2796/774/982/691.

Програма на **Python**

```
n=int(input())
for i in range(n):
  s=input()
  if '+' in s:
      a=s.split('+')
      print(int(a[0])+int(a[1]))
  else:
      a=s.split('-')
      print(int(a[0])-int(a[1]))
```

3.4.26. e1000 Задача A + B

Знайдіть A + B.

Вхідні дані

У кожному рядку задано два цілих числа A та B ($|A|, |B| \leq 30000$). Дані зчитуйте до кінця файлу.

Вихідні дані

Для кожного наведеного прикладу виведіть суму A + B у окремому рядку.

Input	Output
1 1	2
1 2	3

Складність: 33% — 24353/5697/6918/4642.

Програма на **Python** (28 ms, 5.5 MiB)

```
i,o=open('input.txt'),open('output.txt','w')
for line in i: o.write(str(sum(map(int,line.split())))\
  + '\n')
```

Програма на **Pascal**

```
var a,b:longint;
begin
  assign(input,'input.txt');reset(input);
  assign(output,'output.txt');rewrite(output);
  while not eof do
    begin
      readln(a,b);
      writeln(a+b);
    end;
end.
```

3.4.27. e7429 Довга арифметика

Знайти результат додавання або віднімання двох довгих чисел.

Вхідні дані

У першому рядку міститься одне число довжиною не більше 255 знаків. У другому рядку міститься операція: + додавання - віднімання Третій рядок містить друге число, також довжиною не більше 255 знаків.

Вихідні дані

Вивести єдине число — результат виконання заданої операції над цими двома числами.

Input	Output
231211336 + 967047652	1198258988

Складність: 38% — 808/141/206/127.

Програма на **Python** (36 ms, 5,4 MiB)

```
a,o,b=int(input()),input().strip(),int(input())
print(a+b) if o == '+' else print(a-b)
```

3.4.28. e4755 З десяткової у тринадцяткову

Задано число n у десятковій системі числення. Переведіть це число у тринадцяткову систему числення.

Вхідні дані Одне число n ($1 \leq n \leq 1000$).

Вихідні дані Виведіть число n у тринадцятковій системі числення.

Складність: 9% — 511/264/255/233.

Програма на **Python** (30 ms, 5,5 MiB)

```
n,d,r=int(input()),'0123456789ABC',''
while n>0: r=d[n%13]+r; n//=13
print(r)
```

3.4.29. e0441 Найбільш кругле число

Назвемо число більш круглішим, ніж інші числа, якщо воно має більше заключних нулів. Якщо два числа мають однакову кількість заключних нулів, то більш круглішим вважається менше число.

Вхідні дані

У першому рядку вхідних даних задано кількість чисел N ($1 \leq N \leq 100$). Кожен з наступних N рядків містить одне число в межах від 1 до 10^9.

Вихідні дані Вивести найбільш кругле число серед заданих N чисел.

Input	**Output**
4	300
71200	
10	
300	
10001	

Складність: 14% — 2343/965/851/736.

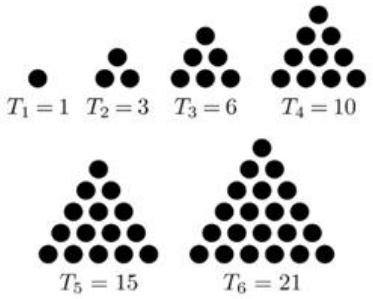
Програма на **Python** (22 ms, 5,5 MiB)

```
n,o,d=int(input()),[],1
for i in range(n): o.append(int(input()))
while o.count(0)<len(o):
```

```
    m=min(o)
    d*=10
    for i in range(len(o)):
        if o[i]//d*d!=o[i]: o[i]=0
    while o.count(0): o.remove(0)
print(m)
```

3.4.30. e6008 Зворотні трикутні числа

Трикутним називається число, що може бути представлене множиною точок, упакованих в рівносторонній трикутник з n точок на стороні. Далі наведено приклади трикутників з відповідними трикутними числами:



Легко бачити, що трикутне число є адитивним варіантом факторіалу:

$$T_n = \sum_{k=1}^{n} k$$

За заданим числом визначіть чи є воно трикутним. Якщо відповідь позитивна, визначіть кількість точок на його стороні.

Приклад: Якщо k = 10, то воно є трикутним з 4-ма точками на стороні, так як $10 = 4 + 3 + 2 + 1$. Якщо k = 11, то воно не є трикутним.

Вхідні дані

Складаються з кількох рядків. Кожен рядок містить ціле число n ($0 < n < 10^9$) без ведучих нулів. Останній рядок містить -1 і не обробляється.

Вихідні дані

Визначіть, чи є значення n трикутним числом. Якщо відповідь позитивна, вивести в рядку кількість точок на стороні. Якщо відповідь негативна, вивести рядок «bad».

Input	**Output**
55	Case 1: 10
1	Case 2: 1
91	Case 3: 13
587	Case 4: bad
499500	Case 5: 999
-1	

Складність: 9% — 647/291/264/241.

Програма на **Python** (32 ms, 5,5 MiB)

```
i=0
while 1:
    T=int(input())
    if T<1: break
    i+=1
    print('Case ',i,': ',sep='',end='')
    k=int((2*T)**.5)
    print(k) if k*(k+1) == 2*T else print('bad')
```

3.4.31. e8865 Однакова парність, e6278 Номери будинків

На вході програми маємо два цілих числа **n** і **m**, записаних в одному рядку через пропуск. Програма повинна вивести **1**, якщо числа **n** і **m** мають однакову парність (тобто одночасно парні або одночасно непарні) і **0** у протилежному випадку.

Вхідні дані Два цілих числа, записаних в одному рядку.

Вихідні дані Відповідь до задачі.

Джерело Серія задач "Абетка програмування"

Складність: 25% — 2679/689/829/618.

Сума чисел з однаковою парністю є парним числом, з різною парністю — непарним. Як раніше вказувалось парність визначається визначається за залишком ділення на 2.

На **Python** (22 ms, 5.1 MiB)

```
print((sum(map(int, input().split())) + 1) % 2)
```

e6278 Номери будинків З'ясувати, чи знаходяться будинки з номерами n та m на одній стороні вулиці.

Вхідні дані Значення n та m ($1 \le n, m \le 100$).

Вихідні дані Вивести 1, якщо будинки з номерами n та m знаходяться на одній стороні вулиці та 0 у протилежному випадку.

На одній стороні вулиці знаходяться будинки з однаковою парністю.

Деякі розв'язки є в [16].

3.4.32. e8866 Подільність

На вході програми маємо два цілих числа **n** і **m**, записаних в одному рядку через пропуск. Програма повинна вивести **1**, якщо число **n** ділиться на **m** націло, і **0** у протилежному випадку.

Вхідні дані Два цілих числа, записаних в одному рядку.

Вихідні дані Відповідь до задачі.

Джерело Серія задач "Абетка програмування"

Складність: 14% — 1602/749/799/684.

На **Python** (20 ms, 5.1 MiB)

```
n, m = map(int, input().split())
print(0) if n % m else print(1)
```

3.4.33. e4736 Чи ділиться на 11?

Для введеного числа перевірити, чи ділиться воно на 11.

Вхідні дані У першому рядку вводиться єдине число.

Вихідні дані У першому рядку виведіть Yes, якщо число ділиться на 11, інакше виведіть No.

Складність: 34% — 4957/1091/1456/956.

Просто поділимо.

На **Python** (23 ms, 5.1 MiB)

```
print('No') if int(input())%11 else print('Yes')
```

Розв'язок є і в [16].

3.4.34. e4756 Остання цифра

Задано число N у десятковій системі числення.

Обчисліть, скільки існує систем числення, у яких число N завершується цифрою k.

Вхідні дані

У вхідному файлі задано число N ($0 \le N \le 10^6$) та цифра k ($0 \le k \le 9$).

Вихідні дані

У вихідний файл виведіть єдине число — кількість систем числення, у яких число N завершується цифрою k. Виведіть 1 якщо таких систем числення нескінченна кількість.

Складність: 32% — 438/65/93/63.

Python (29 ms, 7.7 MiB)

```
n,k = map(int,input().split())
if n == k or k == 0: i=-1
else:
    i=0
    for r in range(2,n):
        if n%r == k: i += 1
print(i)
```

3.4.35. e9428 Дроби: мінімум и максимум

Задано два дроби. Знайдіть їх мінімум та максимум.

Вхідні дані Два дроби a/b и c/d. Всі числа натуральні і не більше 10^9.

Вихідні дані Виведіть спочатку мінімальний, а потім максимальний дріб.

Складність: 36% — 304/87/111/71.

Не зважаючи та те, що маємо справу з дробами (додатними, раціональними числами), як вказувалось раніше потрібно працювати з цілими числами. Тому для порівняння дробів зведемо їх до спільного знаменника та будемо порівнювати чисельники.

На **Python** (20 ms, 5.1 MiB)

```
F,f = input().split()
a,b = map(int,F.split('/'))
c,d = map(int,f.split('/'))
print(f,F) if a*d>b*c else print(F,f)
```

3.4.36. e7401 Друзі Степана

Степан повернувся з міжнародної олімпіади школярів з програмування (IOI) і привіз з собою n різнокольорових каменів в якості сувенірів. Степан зовсім не жадний хлопчик, тому вирішив поділитися камінням зі своїми друзями. Кожному другу Степан віддав рівно один камінь. Виявилося, що у самого Степана залишився теж тільки один камінь. Визначте, скільки ж у нього друзів?

Вхідні дані Одне число n ($1 \le n \le 100$).

Вихідні дані Виведіть одне число — кількість друзів Степана.

Пояснення до прикладу: Степан привіз 2 каменя, один з яких залишився у нього. Отже, другий камінь Степан віддав своєму єдиному другу.

Складність: 3% — 12625/8672/6909/6721.

Очевидно, що кількість друзів $n - 1$.

Програма на **Python**

```
print(int(input())-1)
```

3.4.37. e7336 Пиріжки

Пиріжок у шкільній їдальні коштує a гривень та b копійок. Знайдіть скільки гривень та копійок заплатить Петрик за n пиріжків.

Вхідні дані Три натуральних числа a, b, n $(0 \leq a, b, n \leq 100)$.

Вихідні дані Через пропуск два числа: вартість покупки у гривнях та копійках.

Джерело II етап Всеукраїнської олімпіади в Житомирської області

Складність: 17% — 14834/5167/5043/4194.

Переводимо в копійки. Працюємо з цілими числами.

Програма на **Pascal**

```
var a,b,n:byte;
begin
  read(a,b,n);
  writeln(a*n+b*n div 100,' ',b*n mod 100)
end.
```

Програма на **Python**

```
a,b,n = map(int,input().split())
print(a*n+b*n//100,b*n%100)
```

Деякі розв'язки є і в [16].

3.4.38. e0126 Номер квартири

Багатоквартирний будинок з N квартир має P під'їздів і Q поверхів, причому на кожному поверсі кожного під'їзду розміщено однакову

кількість квартир. Визначити в якому під'їзді та на якому поверсі знаходиться квартира з заданим номером K.

Вхідні дані В єдиному рядку файлу записано значення N, P, Q, K. $1 \le K \le N \le 1000, P * Q \le N$.

Вихідні дані В єдиний рядок вихідного файлу треба вивести номер під'їзду і поверх, на якому знаходиться квартира з номером K.

Джерело II етап Всеукраїнської олімпіади з інформатики в Житомирській обл.

Складність: 21% — 6834/2223/2453/1942.

Python (30 ms, 5.5 MiB)

```
n,p,q,k=map(int, input().split())
print((k-1)*p//n+1, (k-1)*p*q//n%q+1)
```

Прості розрахунки

3.4.39. a0312 Арифметична прогресія

Задані перший та другий елементи арифметичної прогресії. Потрібно написати програму, яка розраховує елемент прогресії за його номером

Вхідні дані

Вхідний файл INPUT.TXT містить три цілих числа, розділених пробілами — перший елемент прогресії $a1$ ($1 \le a1 \le 1000$), другий елемент прогресії $a2$ ($1 \le a2 \le 1000$), и номер потрібного елемента n ($1 \le n \le 1000$).

Вихідні дані

Вихідний файл OUTPUT.TXT має містити одне ціле число — n-й елемент арифметичної прогресії.

Складність задачі: 15%, розв'язуваність 93% (21113)

Використовуємо, посилаємось на 2.1.3.

Pascal (0.03 с, 1308 Кб)

```
var n,a1,a2: integer;
begin
  readln(a1,a2,n);
  write(a1+(n-1)*(a2-a1));
end.
```

3.4.40. e8889 Кількість непарних цифр

Задано п'ятизначне натуральне число **n**. Потрібно знайти кількість непарних цифр в числі **n**.

Вхідні дані Задане п'ятизначне натуральне число.

Вихідні дані Відповідь до задачі.

Джерело Серія задач "Абетка програмування"

Складність: 18% — 958/425/474/388.

Для запобігання некоректного введення, після введення переведемо в ціле число, а потім знову переведемо в рядок. Розкладемо на символи-цифри. Замість циклу використаємо генератор. Переведемо кожну символ-цифру в число і поділимо за модулем 2 щоб виділити непарні цифри (для них результат 1, для парних — 0). Підсумуємо по всім цифрам, виведемо результат.

На **Python** (21 ms, 5.1 MiB)

```
print(sum([int(d)%2 for d in list(str(int(input())))]))
```

3.4.41. e8909 Довжина послідовності

На вході програми маємо послідовність цілих чисел, що закінчується числом 0. Потрібно знайти довжину даної послідовності, не враховуючи останнього нуля.

Вхідні дані Послідовність цілих чисел, по одному числу в кожному рядку.

Вихідні дані Одне число — довжина даної послідовності.

Джерело Серія задач "Абетка програмування"

Складність: 8% — 1073/567/544/501.

На **Python** (19 ms, 5.1 MiB)

```
s = 0
while 1:
    n = int(input())
    if not n: break
    s += 1
print(s))
```

3.4.42. e8913 Кількість непарних

На вході програми маємо послідовність цілих чисел, що закінчується числом 0. Потрібно знайти кількість непарних чисел в даній послідовності.

Вхідні дані Послідовність цілих чисел, по одному числу в кожному рядку.

Вихідні дані Одне число — кількість непарних чисел в послідовності.

Джерело Серія задач "Абетка програмування"

Складність: 4% — 609/352/340/328.

На **Python** (19 ms, 5.1 MiB)

```
s = 0
while 1:
    n = int(input())
    if not n: break
    s += n % 2
print(s)
```

3.4.43. c004A Кавун

В один з жарких літніх днів Петя і його друг Вася вирішили купити кавун. На їх погляд вони вибрали найбільший і стиглий. Після недовгої процедури зважування ваги показали w кілограм. Поспішно прибігши додому, знемагаючи від спраги, хлопці почали ділити придбану ягоду, однак перед ними постало нелегке завдання. Петя і Вася є великими шанувальниками парних чисел, тому хочуть поділити кавун так, щоб частка кожного важила саме парне число кілограм, при цьому не обов'язково, щоб частки були рівними за величиною. Хлопці дуже сильно втомилися і хочуть швидше приступити до трапези, тому Ви повинні підказати їм, чи вдасться поділити кавун, враховуючи їх побажання. Зрозуміло, кожному повинен дістатися шматок додатної ваги.

Вхідні дані В першому і єдиному рядку вхідних даних записано ціле число w ($1 \leq w \leq 100$) — вага купленого хлопцями кавуна.

Вихідні дані

Виведіть YES, якщо хлопці зможуть поділити кавун на дві частини, кожна з яких важить парне число кілограм, и NO в іншому випадку.

Складність: 1200, розв'язали 131863.

Вага парних шматків може бути описана $2n$ ($n \geq 1$) та $2k$ ($k \geq 1$). Таким чином вага $w = 2(n + k)$ ($(n + k) \geq 2$) має бути парною більше 2.

На **Python** (24 ms, 5.1 MiB)

```
w = int(input())
print('NO') if w%2 or w == 2 else print('YES')
```

На [10] є ще цікаві розв'язки, наведені нижче

```
i=int(input())
print(['YES','NO'][i%2 or i<3])
```

```
print(["NO","YES"][(-2)**int(input())>4])
```

```
print("YNEOS"[2**int(input())%24<9::2])
```

найбільш цікавим і інтуїтивно зрозумілим напевно є

```
print(["YES","NO"][max(int(input()),3)&1])
```

3.4.44. e0955 Квадрат суми

Знайдіть квадрат суми цифр чотирицифрового натурального числа.
Вхідні дані Одне натуральне чотирицифрове число.
Вихідні дані Виведіть квадрат суми цифр заданого числа.
Складність: 3% — 5755/3853/3570/3451.

Використовуємо, спираємось на 3.4.13

Python (20 ms, 5.1 MiB)

```
print(sum(map(int,list(input())))**2)
```

Деякий розв'язок є в [16].

3.4.45. e0953 Остача

Знайти остачу від ділення останньої цифри на першу в даному натуральному трицифровому числі.
Вхідні дані Одне натуральне трицифрове число.
Вихідні дані Вивести шукану остачу.
Складність: 6% — 6815/3899/3603/3400.

На **Python**

```
n=input()
print(int(n)%10%int(n[0]))
```

3.4.46. e1008 Системи числення

Дано ціле невід'ємне число у m-й системі числення. Потрібно вивести це число в k-й системі числення.

Вхідні дані

Вхідний файл у першому рядку містить два числа m і k (у десятковій системі числення), у другому рядку — число для переведення. $2 \le m, k \le 36$, для представлення цифр 10...35 використовуються прописні латинські літери $A...Z$ відповідно, число розрядів заданого числа не перевищує 1000.

Вихідні дані

У вихідний файл виведіть шукане число без ведучих нулів.

Input	**Output**
10 36 29234652	HELLO

Складність: 42% — 1320/240/325/190.

На **Python**

```
d='0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ'
m,k=map(int,input().split())
n,r=int(input(),m),''
if n==0: r='0'
while n>0: r=d[n%k]+r; n//=k
print(r)
```

3.4.47. e0057 Метелик-санітар

Учні, йдучи з дому до школи або навпаки — зі школи додому, полюбляють їсти цукерки. Але, як завжди, ця приємна справа інколи має неприємні наслідки — дітки іноді викидають обгортки на шкільному подвір'ї.

Мурзик завжди слідкував за чистотою шкільного двору і йому у цьому з радістю допомагали метелики, вдячні за чудові фотографії, зроблені ним. Метелики могли використовувати власні крильця як лінзи, причому вони могли змінювати їх фокусну відстань. Помітивши обгортку від цукерки, що лежала на шкільному подвір'ї у точці з координатами

X1, Y1, метелик перелітав у точку з координатами X2, Y2, Z2, розташовану на шляху сонячного проміння до обгортки і, змінюючи фокусну відстань своїх крилець-лінз, спалював обгортку від цукерки.

Яку оптичну силу D мали крильця-лінзи метелика у цей момент?

Вхідні дані У першому рядку 2 числа: координати X1, Y1 обгортки від цукерки. У другому — 3 числа: координати X2, Y2, Z2 метелика у момент спалювання обгортки.

Вихідні дані Всі вхідні дані цілі числа, що не перевищують за модулем 1000.

Input	**Output**
10 36 29234652	HELLO

Автор Анатолій Присяжнюк

Складність: 10% — 7297/3541/3535/3185.

Знаємо, що оптична сила — $1/f$, обгортки — в фокусі, оскільки Сонце на ∞.

На **Python**

```
x,y = map(int, input().split())
X,Y,Z = map(int, input().split())
print('%.3f'%(1/((X-x)**2+(Y-y)**2+Z**2)**.5)))
```

3.4.48. e9637 Діно та висотки

Діно подорожує Китаєм (у той час ще не був розповсюджений Короновірус) і бачить там висотні будинки в кількості 10^9. Висота першого будинку 1, другого 1+2, третього 1+2+3 і т. д. Висота останньої висотки $1 + 2 + 3 + ... + 10^9$ (всі величини вказані у метрах).

Діно після мандрівки заснув, а вранці побачив неймовірний пейзаж: вночі падав сніг і висоти усіх висоток збільшились на однакову величину! Діно бажає знати скільки метрів снігу випало. Для цього він підійшов до двох сусідніх будинків (наприклад до 3-го та 4-го) і виміряв їх нові значення висот. Допоможіть Діно визначити скільки метрів снігу випало, якщо висоти цих висоток рівні відповідно a та b метрів.

Вхідні дані В єдиному рядку записані цілі числа a і b ($a \leq b$). Гарантовано, що правильна відповідь завжди існує, а висота снігу, що випав, не більша ніж 10^9 метрів.

Вихідні дані Виведіть висоту снігу, що випав.

Джерело

Півфінал Республіканської олімпіади Азербайджану 2019-2020

Складність: 23% — 547/191/200/155.

Висоти будинків є сумою арифметичної прогресії (2.1.3) — висота n-го $a = \frac{n(n+1)}{2}$, наступного $(n+1)$-го — $b = \frac{(n+1)(n+2)}{2}$. За умовою задачі різниця висот будинків не залежить від снігу, що дозволяє визначити номер будинку n. Тоді знатимемо і висоту снігу на ньому.

На **Python** (21 ms, 5.1 MiB)

```
a,b=map(int,input().split())
print(a-(b-a-1)*(b-a)//2)
```

Факторіал

3.4.49. a0018 Факторіал, e0271 Факторіал! e1658 Факторіал

a0018 Факторіал. Потрібно вирахувати факторіал цілого числа N. Факторіал позначають як $N!$ і розраховують за формулою: $N! = 1 * 2 * 3 * ... * (N-1) * N$, причому $0! = 1$.

Так же допустимо рекурентне співвідношення: $N! = (N-1)! * N$

Вхідні дані В єдиному рядку вхідного файлу INPUT.TXT записано одне ціле невід'ємне число N ($N < 1000$).

Вихідні дані

У вихідному файлі OUTPUT.TXT потрібно вивести одне ціле число — значення $N!$.

Складність задачі: 42%, розв'язуваність 66% (9670)

Використовуємо, посилаємось на **??**.

Pascal (0.03 с, 1308 Кб)

```
var n,p,prns,i : longint;
    f : array [1..650] of integer;
    nf,j : integer;
begin
  read(n);
  if (n=0) or (n=1) then begin write(1); exit; end;
  for j:=1 to 650 do f[j]:=0; f[1]:=1; nf:=1;
   for i:=2 to n do
    begin prns:=0;
```

```
        for j:=1 to nf do
          begin
            p:=f[j]*i+prns;
            f[j]:=p mod 10000;
            prns:=p div 10000;
          end;
        if prns>0 then
          begin inc(nf); f[nf]:=prns; prns:=0; end;
      end;
    write(f[nf]);
    for i:=nf-1 downto 1 do begin
      case f[i] of
       0   ..   9 : write('000');
       10  ..  99 : write('00');
       100..999 : write(0)
      end;
     write(f[i]);
    end;
end.
```

Python (0.031 с, 698 Кб)

```
from math import factorial
print(factorial(int(input())))
```

Деякий розв'язок є в [16].

Аналогічною є умова **e0271 Факторіал!**.
Відмінність тільки в $(0 \le n \le 3000)$.
Складність: 41% — 10623/2328/3042/1809.
Розв'язок наведено вище — Python (24 ms, 7.7 MiB).

Також **e1658 Факторіал**. $(0 \le n \le 20)$.
Складність: 12% — 10477/3778/2862/2523.

3.4.50. e1214 Мультифакторіал

k-мультифакторіалом числа n називається добуток усіх додатних чисел вигляду $n - k \cdot x, x = 0, 1, 2, ...$ та позначається $fac_k(n)$.

Наведемо формальне визначення мультифакторіала:
$fac_k(n) = n$, якщо $k \ge n$;
$fac_k(n) = n \cdot fac_k(n-k)$, якщо $k < n$;

За заданими n та k необхідно обчислити $fac_k(n)$. Якщо результат буде більшим за 10^{18}, то слід надрукувати "overflow".

Вхідні дані Два цілі числа n та k ($1 \le n, k \le 2 \cdot 10^9$).

Вихідні дані Вивести значення $fac_k(n)$. Якщо воно строго більше за 10^{18}, то вивести "overflow".

Input #1	**Output #1**
14 3	12320
Input #2	**Output #2**
1000 2	overflow

Складність: 30% — 5435/800/899/630.

Згідно означення домножуємо попередній результат до тих пір поки не отримаємо результат або переповнення (досягнемо заданої верхньої межі).

Python (32 ms, 5.5 MiB)

```
n,k=map(int,input().split())
f=1
for i in range(n,1,-k):
    f *= i
    if f>10**18: print('overflow'); exit()
print(f)
```

3.4.51. e5900 Мультифакторіал

У даній задачі вам потрібно обчислити m-кратний факторіал числа k. Нагадаємо як обчислювати m-кратний факторіал. Позначимо його F. Нехай $t_i = k - mi$. Вірною є наступна формула:

$$F = \prod_{i=0\ldots\infty, t_i>0} t_i$$

Вхідні дані У першому рядку записано два цілих невід'ємних числа: m та k ($1 \le m \le 100$, $1 \le k \le 50000$).

Вихідні дані У єдиному рядку вихідних даних вивести m-кратний факторіал числа k. Не виводьте ведучих нулів.

Ліміт часу 4 с

Автор Євген Соболєв

Джерело Літня школа Севастополь 2013, Хвиля 1, День 5

Складність: 43% — 46/13/14/8.

Перемножаємо згідно означення.

Python (1885 ms, 5.9 MiB)

```
m,k = map(int,input().split())
f=1
for i in range(k,1,-m): f *= i
print(f)
```

3.4.52. e0062 Факторіал

За значенням $n!$ ($n! = 1 \cdot 2 \cdot ... \cdot (n-1) \cdot n$) визначити значення n.

Вхідні дані Значення $n!$ ($1 \leq n \leq 2000$).

Вихідні дані Вивести натуральне число n.

Автор Сергій Матвійчук

Джерело III етап Всеукраїнської олімпіади з інформатики в Житомирській обл. 1999-2000

Складність: 52% — 9791/1461/2643/1280.

Введене число ділимо послідовно на натуральні числа 2,3,... до тих пір доки не отримаємо 1. Останній дільник і є шуканою величиною.

Pascal

```
var s,t:ansistring;
    i,j,k,r,d:word;
    n:int64;
begin
    read(s);
    if s='1' then begin writeln(1); exit end;
    for j:=2 to 2000 do
      begin
        if length(s)<19 then
          begin
            val(Copy(s,1,length(s)),n);
            n:=n div j;
            str(n,s);
            if n=1 then break;
          end
        else
          begin
            val(Copy(s,1,18),n);
```

```
            r:=n mod j;
            n:=n div j;
            str(n,t);
            k:=length(t);
            setlength(t,k+length(s)-18);
            for i:=19 to length(s) do
              begin
                d:=r*10+ord(s[i])-48;
                r:=d mod j;
                t[k+i-18]:=chr((d div j)+48);
              end;
            s:=t;
          end
      end;
    writeln(j)
end.
```

Python

```
F,n,f = int(input()),1,1
while f<F: n += 1; f *= n
print(n)
```

Розв'язок є і в [16].

3.4.53. e7441 Факторіал

Вхідні дані Одне ціле число $n (1 \le n \le 10^{18})$.
Вихідні дані Вивести $n!$ (mod 3469708049238200000).
Джерело 2014 KBTU Open, Весна Казахстан, Алма-Ата, Задача D
Складність: 72% — 1021/84/247/62.

Знайдемо довжину числа 3469708049238200000 — 19. Таким чином робимо висновок, що не виходимо за рамки звичайної процесорної арифметики, довга арифметика нам зараз не потрібна. Визначимо число n факторіал якого має не менше 19 нулів **??**. Далі просто порахуємо.

Python (29 ms, 5.4 MiB)

```
import math
n=int(input())
print(0) if n>55 else print(math.factorial(n) %\
    3469708049238200000)
```

3.4.54. e0149 Факторіал - 2

Знайти кількість цифр у запису факторіалу натурального числа n. (Факторіал числа n — це добуток усіх натуральних чисел від 1 до n).

Вхідні дані Одне число n ($1 \leq n \leq 1000000$).

Вихідні дані Вивести кількість цифр у числі $n!$.

Автор Сергій Матвійчук

Джерело III етап Всеукраїнської олімпіади з інформатики в Житомирській обл. 2008-2009 р

Складність: 45% — 3604/755/1265/690.

Python (240 ms, 5.6 MiB)

```
import math
n,f,k = int(input()),1.,0
for i in range(1,n+1):
    f *= i
    if f>1e300:
        f /= 1e300
        k += 1
print(int(math.log(f,10)+1e-2)+1+300*k)
```

Римські числа

3.4.55. e4103 Римські числа

В ході роботи над Вашим новим проектом виникла необхідність оперувати римськими числами. Перед Вами стоїть задача конвертації числа з римської системи числення в десяткову.

Все числа в рамках Вашого проекту можна записати за допомогою 7 цифр: I = 1; V = 5; X = 10; L = 50; C = 100; D = 500; M = 1000.

Натуральні числа записуються за допомогою повторення цих цифр. При цьому, якщо більша цифра стоїть перед меншою, або рівною, то вони складаються, якщо ж менша – перед більшою, то менша віднімається з більшої.

Вхідні дані

В першому рядку написано число N ($1 \leq N \leq 100$) — кількість римських чисел, які потрібно конвертувати. В наступних N рядках записані самі римські числа, по одному в кожному рядку.

Вихідні дані

Необхідно вивести N чисел в десятковій системі числення, які є результатом конвертації вхідних римських чисел. Послідовність десяткових чисел має відповідати послідовності відповідних римських чисел. Гарантується, що конвертоване число знаходиться в інтервалі [1, 1000].

Input	**Output**
4	9
IX	31
XXXI	46
XLVI	888
DCCCLXXXVIII	

Складність: 11% — 301/180/149/132.

Програма на **Python** (42 ms, 5,5 MiB)

```
r=['M','CM','D','CD','C','XC','L','XL','X','IX','V',\
   'IV','I']
d=[1000,900,500,400,100,90,50,40,10,9,5,4,1]
n=int(input())
for k in range(n):
    a,s=input(),0
    for j in range(2):
        for i in range(13):
            while a.find(r[i]) == 0:
                s += d[i]
                a = a.replace(r[i],'',1)
    print(s)
```

3.4.56. e0007 Римські числа

Підрахувати суму двох натуральних чисел A і B, записаних в римській системі числення. Відповідь записати також, в римській системі числення.

M = 1000, D = 500, C = 100, L = 50, X = 10, V = 5, I = 1 (Всі числа — менші 2000).

Вхідні дані

В рядку записано два числа римською системою числення, між якими стоїть знак + .

Вихідні дані

Одне число, сума чисел також римською системою числення. Числа в римській системі числення записано великими латинськими літерами.

Input	**Output**
III+IV	VII

Складність: 29% — 8307/2532/3251/2299.

Вводимо вхідний вираз. Це рядок. Перетворюємо кожну римську цифру цього рядка в число (ціле десяткове) та підсумовуємо в результат. Після перетворення видаляємо римську цифру, а в кінці і знак плюс. Завершення циклу — порожній рядок. По завершенню робимо зворотнє перетворення результату в римську систему числення.

Програма на **Python**

```
r=['M','CM','D','CD','C','XC','L','XL','X','IX','V',
   'IV','I']
d=[1000,900,500,400,100,90,50,40,10,9,5,4,1]
a,s,n=input(),0,''
for j in range(2):
    for i in range(13):
        while a.find(r[i])==0:
            s+=d[i]
            a=a.replace(r[i],'',1)
    a=a.replace('+','')
for i in range(13):
    for j in range(3):
        if s>=d[i]:
            n+=r[i]
            s-=d[i]
print(n)
```

Фібоначчі

3.4.57. e4730 Фібоначчі

Числа Фібоначчі — це послідовність чисел F(n), яка задається формулою: F(0)=1, F(1)=1, F(n)=F(n-1)+F(n-2).

За заданим числом n вивести n-те число Фібоначчі.

Вхідні дані

Невід'ємне число n (n≤45) — номер числа Фібоначчі, яке потрібно вивести.

Вихідні дані Вивести n-те число Фібоначчі.

Складність: 14% — 9148/3395/2761/2374.

Створимо список чисел Фібоначчі та виведемо потрібне.

Програма на **Python** (26 ms, 5.5 MiB)

```
n,f = int(input()),[1,1]
for i in range(2,n+1): f.append(f[i-1]+f[i-2])
print(f[n])
```

Максимальна довжина масиву (списку) зовсім невелика (45), але якщо дуже потрібно можна використати всього 4 змінні. Наведемо модифікований розв'язок з [16], що не використовує масиви

```
n,f,f1,f2 = int(input()),1,1,1
if n>1:
    for i in range(n-1): f = f1+f2; f1,f2 = f,f1
print(f)
```

3.4.58. e0192 Просто Фібоначчі

Знайти N-е по порядку просте число Фібоначчі.

У вхідному файлі число N ($1 \le N \le 10$).

До вихідного файлу потрібно записати N-е по порядку просте число Фібоначчі.

Автор Сергій Матвійчук

Складність: 27% — 5541/1461/1750/1278.

За такої кількості вхідних даних окремо розраховуємо результати, заносимо в масив і в програмі виводимо потрібний елемент масиву.

На **Python** (30 ms, 5.4 MiB)

```
print([0,2,3,5,13,89,233,1597,28657,514229,433494437] \
    [int(input())])
```

3.4.59. e1358 Кількість чисел Фібоначчі

Послідовність Фібоначчі — це така послідовність, у якій кожен елемент дорівнює сумі двох попередніх, за винятком перших двох елементів: $F_1 = 1,\ F_2 = 1,\ F_n = F_{n-2} + F_{n-1}$.

1 1 2 3 5 8 13 21 ...

Задано масив цілих чисел. Скільки знаходиться в ньому чисел Фібоначчі?

Вхідні дані

У першому рядку записано кількість вхідних чисел k. У наступному рядку записано k чисел $a_1, a_2, ..., a_k (0 < k \leq 1000,\ 0 \leq a_i < 2^{63})$.

Вихідні дані Вивести одне число — кількість чисел Фібоначчі у заданому масиві.

Складність: 14% — 1156/479/472/407.

Згенеруємо масив чисел Фібоначчі, підрахуємо скільки їх у введеному масиві.

На **Python** (34 ms, 7.7 MiB)

```
f,n=[1,1],0
for i in range(2,93): f.append(f[i-1]+f[i-2])
input()
a=[int(s) for s in input().split()]
for i in a:
    if i in f: n+=1
print(n)
```

Інші задачі з цілими числами

e1607 Число у зворотньому порядку Умова та обробка числа як тексту наведено пізніше в 3.10.2.

e0947 Зворотній порядок Умова та обробка числа як тексту наведено в 3.10.2.

e0943 Перестановка цифр трицифрового Умова та обробка числа як тексту наведено в 3.10.2.

e1608 Число-паліндром Умова та розв'язок наведені в 3.10.3.

e8243 Перша цифра числа Умова та обробка числа як тексту наведено в 3.10.5.

e1605 Друга цифра числа Умова та код програми наведено в 3.10.6.

e0949 Двозначне з чотиризначного Умова та розв'язок наведені в 3.1.5.

e0134 Два кола – 2 Умова та розв'язок наведені в 3.20.37.

3.5. Вбудовані можливості мов

Розглянемо як використання вбудованих можливостей деяких мов програмування спрощує розв'язання задач. Акцент на можливостях, що в більшості інших мов відсутні.

В задачі **e1001** (3.4.12) використана довга арифметика та двійкова система числення (Python).

В задачі **e1607** (3.10.2) для обернення тексту використано зрізи (Python).

В **e1609** (3.10.7) для підрахунку використано count() (Python).

В **e0852** використано вбудовану довгу арифметику Python.

В **a0018** 3.4.49 використовуємо вбудовану функцію розрахунку факторіалу (Python).

В **e8867** 3.5.1, **e8868** 3.5.2, **e8869** 3.5.3, **e8870** 3.5.4, **e8871** 3.5.5, **e0928** 3.13.1, **e9561**3.14.2, **e9563**3.14.4,**e9565**3.14.6 використовуються функції **min** та **max** (Python).

В **e0108**3.6.2, **e9566**3.14.7 використовується сортування (sort(), sorted()) (Python).

В **e0147** 3.21.1 використовується робота з датами (Python).

В **e9531** 3.9.1 використовується робота з комплексними числами (Python).

e2802 Бітове подання Умова та код наведені в 3.12.1.

3.5.1. e8867 Менше з двох, e1357 Кількість нулів, на які закінчується число

На вході програми маємо два цілих числа **a** і **b**, записаних в одному рядку через пропуск. Потрібно вивести менше з них.

Вхідні дані

В єдиному рядку записані через пропуск два цілих числа **a** і **b**.

Вихідні дані Відповідь до задачі.

Джерело Серія задач "Абетка програмування"

Складність: 7% — 1841/994/973/906.

Знаходимо мінімум з чисел.

На **C++**

```
#include <bits/stdc++.h>
using namespace std;
int main() {
        int a,b;
        cin >> a >> b;
        cout << min(a,b) << endl;
}
```

На **Python** (22 ms, 5.1 MiB)

```
print(min(map(int, input().split())))
```

Задача **e1357 Кількість нулів, на які закінчується число**

Деяке натуральне число N можна розкласти на прості множники, серед яких M дільників, що дорівнюють 2 і K — 5. На скільки нулів закінчується натуральне число N?

Вхідні дані У єдиному рядку знаходиться 2 числа M і K ($0 \leq M, K \leq 32000$).

Вихідні дані Одне число — кількість кінцевих нулів.

Складність: 9% — 880/486/454/412.

Кількість нулів в кінці числа — це кількість дільників 10 цього числа. Оскільки $10 = 2 \times 5$? то шукаємо мінімум N та K.

Ця задача має цей же програмний код, наведений вище.

3.5.2. e8868 Більше з двох

На вході програми маємо два цілих числа **a** і **b**, записаних в одному рядку через пропуск. Потрібно вивести більше з них.

Вхідні дані В єдиному рядку записані через пропуск два цілих числа **a** і **b**.

Вихідні дані Відповідь до задачі.

Джерело Серія задач "Абетка програмування"

Складність: 4% — 1172/851/834/798.

Знаходимо максимум з чисел.

На **Python** (20 ms, 5.1 MiB)

```
print(max(map(int, input().split())))
```

3.5.3. e8869 Впорядкування двох

На вході програми маємо два цілих числа **a** і **b**, записаних в одному рядку через пропуск. Задані числа потрібно вивести в порядку зростання тобто спочатку менше, а потім більше з них.

Вхідні дані В єдиному рядку записані через пропуск два цілих числа **a** і **b**.

Вихідні дані В одному рядку спочатку менше, а потім більше з чисел **a** і **b**.

Джерело Серія задач "Абетка програмування"

Складність: 9% — 1824/872/885/807.

На **Python** (23 ms, 5.1 MiB)

```
a,b = map(int,input().split())
print(min(a,b),max(a,b))
```

Альтернативно ті ж (23 ms, 5.1 MiB) але один рядок.

```
print(*(sorted(map(int,input().split()))))
```

3.5.4. e8870 Менше з трьох

На вході програми маємо три цілих числа **a**, **b** і **c**, записаних в одному рядку через пропуск. Потрібно вивести менше з них.

Вхідні дані
В єдиному рядку записані через пропуск три цілих числа **a**, **b** і **c**.
Вихідні дані Відповідь до задачі.
Джерело Серія задач "Абетка програмування"
Складність: 7% — 1465/820/794/737.

Використовуємо, посилаємось на 3.5.1.
Знаходимо мінімум з чисел.

На **Python** (22 ms, 5.1 MiB)

```
print(min(map(int, input().split())))
```

3.5.5. e8871 Більше з трьох

На вході програми маємо три цілих числа **a**, **b** і **c**, записаних в одному рядку через пропуск. Потрібно вивести більше з них.

Вхідні дані
В єдиному рядку записані через пропуск три цілих числа **a**, **b** і **c**.
Вихідні дані Відповідь до задачі.
Джерело Серія задач "Абетка програмування"
Складність: 6% — 1286/838/832/782.

Використовуємо, посилаємось на 3.5.2.
Знаходимо максимум з чисел.

На **Python** (18 ms, 5.1 MiB)

```
print(min(map(int, input().split())))
```

3.5.6. e8872 Впорядкування трьох

На вході програми маємо три цілих числа **a**, **b** і **c**, записаних в одному рядку через пропуск. Задані числа потрібно вивести в порядку зростання.

Вхідні дані
В єдиному рядку записані через пропуск три цілих числа **a**, **b** і **c**.
Вихідні дані Відповідь до задачі.
Джерело Серія задач "Абетка програмування"
Складність: 8% — 1690/761/748/689.

На **Python** (20 ms, 5.1 MiB)

```
print(*sorted(map(int,input().split())))
```

3.5.7. c0112 $a^b - b^a$

Задано два натуральних числа a та b. Знайдіть $a^b - b^a$.
Вхідні дані
Вихідні дані Виведіть відповідь
Обмеження часу: 0,25 с. **Складність**: розв'язали 1201.

На **Python** (93 мс, 8 КБ)

```
a,b = map(int,input().split())
print(a**b-b**a)
```

3.5.8. e4757 Ознака подільності

Число n подано у двійковій системі числення. Визначіть, чи ділиться воно на **15**.

Вхідні дані Одне число **n** (довжина числа не перевищує **10000** двійкових розрядів).

Вихідні дані Виведіть **YES**, якщо число n ділиться на **15**, і **NO** у протилежному випадку.

Складність: 38% — 1228/208/294/181.

Просто виконаємо дію. Використовуємо вбудовану двійкову та довгу арифметику.

Програма на **Python** (35 ms, 7,5 MiB)

```
print('NO') if int(input(),2)%15 else print('YES')
```

3.5.9. e1121 *A^B mod C*

За заданими a, b, c обчисліть значення $a^b \ mod \ c$ ($1 \leq a, b, c < 2^{63}$).

Вхідні дані Складається з декількох тестів. Кожен тест задається в одному рядку та містить три числа a, b та c.

Вихідні дані Для кожного тесту в окремому рядку вивести результат виконання операції $a^b mod \ c$.

Джерело II етап Всеукраїнської олімпіади 2010-2011 м.Бердичів

Складність: 56% — 4644/555/827/365.

Використаємо вбудовану функцію pow(a,b,c), яка виконує потрібну дію.

На **Python** (29 ms, 5.4 MiB)

```
for line in open('input.txt'):
    a,b,c = map(int,line.split())
    print(pow(a,b,c))
```

3.5.10. e5322 Системи числення – 1

Перевести число a, записане у двійковій системі числення, у шістнадцяткову. Вивести число a у шістнадцятковій системі числення без ведучих нулів.

Вхідні дані Число, записане у двійковій системі числення, 0 < довжина числа $\leq 10^4$.

Вихідні дані Виведіть число, переведене у шістнадцяткову систему числення, записане за допомогою символів '0', ..., '9' та 'A', ..., 'F'.

Складність: 25% — 704/218/255/192.

Python (37 ms, 7.5 MiB)

```
print((hex(int(input(),2)))[2:].upper())
```

3.5.11. e5320 Доповнювальний код – 1

Напишіть програму, яка за заданими числами A та n записує подання числа A у n-розрядному двійковому доповнювальному коді.

Перший рядок вхідних даних містить число A, другий рядок — число n, при цьому $2 \leq n \leq 16, -2^{n-1} \leq A \leq 2^{n-1} - 1$.

Програма повинна вивести рядок з n символів, який містить запис числа A у n-розрядному двійковому доповнювальному коді, перший символ — старший знаковий розряд.

Вхідні дані

У єдиному рядку — два числа, A та n. $2 \leq n \leq 16, -2^{n-1} \leq A \leq 2^{n-1} - 1$.

Вихідні дані

Виведіть число у n-розрядному доповнювальному коді.

Input 1	**Output 1**
5 8	00000101
Input 2	**Output 2**
-5 8	11111011

Складність: 17% — 1176/503/506/422.

Python (26 ms, 7.7 MiB)

```
a,n = map(int,input().split())
print(bin((1<<n)+a)[2:]) if a<0 else print('0'* \
    (n-len(a)+2)+bin(a)[2:])
```

3.5.12. e5321 Доповнювальний код – 2

Дано запис деякого числа у двійковому доповнювальному коді. Виведіть десятковий запис цього числа.

Вхідні дані

Рядок S ($2 \leq |S| \leq 16$), що містить послідовність з 0 та 1.

Вихідні дані Виведіть число в десятковому запису.

Input 1	**Output 1**
00000101	5
Input 2	**Output 2**
11111011	-5

Складність: 28% — 287/99/131/94.

Крім int(,2), що переводить з двійкової системи, використовуємо також 3.12.2.

Python (26 ms, 7.5 MiB)

```
s=input()
print(int(s,2)) if s[0]=='0' else print(int(s,2)- \
    (1<<len(s)))
```

3.5.13. e7339 Послідовність

Знайти N-й член послідовності 1 10 11 100 101 110 111 1000 .
Вхідні дані Одне натуральне число N ($N \leq 10000$).
Вихідні дані N-й член послідовності.
Складність: 17% — 2153/991/938/780.

Легко побачити, що послідовність представляє собою послідовність натуральних чисел в двійковій системі числення. Тому переводимо введене число в двійкову систему і видаляємо префікс.

Програма на **Python** (28 ms, 5,5 MiB)

```
print(bin(int(input()))[2:])
```

Розв'язки на C++ є в [16].

3.5.14. e2674 Скорочення дробу

Скоротити заданий дріб.
Вхідні дані Чисельник та знаменник дробу (цілі числа, за модулем не перевищують 10^9).
Вихідні дані Вивести чисельник та знаменник скороченого дробу.
Складність: 31% — 5079/1042/1154/800.

Знайдемо НСД і скоротимо чисельник та знаменник.

На **Python**

```
n,d=map(int,input().split())
a,b=abs(n),abs(d)
while a*b>0:
    if a>b: a%=b
    else: b%=a
print(n//(a+b),d//(a+b))
```

```
from math import gcd
n,d = map(int,input().split())
print(n//gcd(n,d),d//gcd(n,d))
```

На **Python** (38 ms, 6.5 MiB) Вбудовані можливості з роботи з дробами

```
from fractions import Fraction
n,d = map(int,input().split())
f = Fraction(n,d)
```

```
print(f.numerator,f.denominator)
```

3.6. Інші прості розрахунки

3.6.1. e8254 Номера готелю

Готель має n поверхів. Лобі, ресторан і тренажерний зал розміщений на першому поверсі. Номери знаходяться з 2-го по n-ий поверхи. На кожному поверсі розміщено m стандартних номерів. Якщо кожен стандартний номер містить 3 гостя, Яку найбільшу кількість гостей може помістится у всіх стандартних номерах готелю?

Вхідні дані Два натуральних числа n і m ($n, m \leq 10^6$).

Вихідні дані Виведіть найбільшу кількість гостей, яку можна помістити у всіх стандартних номерах готелю.

Автор Михайло Медведєв

Складність: 12% — 6119/2300/2248/1973.

Python (23 ms, 7.7 MiB)

```
n,m=map(int,input().split())
print((n-1)*m*3)
```

3.6.2. e0108 Середнє з чисел

Дано три різні числа **a**, **b**, **c**. Вивести середнє з них.

Вхідні дані Числа **a**, **b**, **c** цілі та за модулем не перевищують 1000.

Вихідні дані Вивести середнє з трьох чисел.

Джерело II етап Всеукраїнської олімпіади в Житомирській області

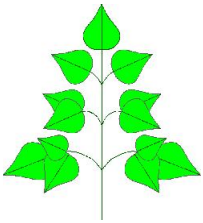
Складність: 15% — 38615/13090/12386/10515.

На **Python** (19 ms, 5.1 MiB)

```
print(sorted(map(int,input().split()))[1])
```

Деякий розв'язок є і в [16].

3.6.3. e0248 Юний садівник



Мама попросила Васю полити всі молоді деревця у саду. Вася знає, що поки дерева маленькі, їх потрібно дуже добре поливати. А ось скільки поливати — невідомо. Але Вася — дуже розумний хлопчик. Він уважно прочитав весь підручник ботаніки для середньої школи і вияснив, що полив прямо пропорційний кількості листочків на дереві. Для гарного росту дерев достатньо виливати під дерево щоденно по одному літру води для кожного листка.

На щастя Васі виявилось, що листки на деревах ростуть ярусами, причому на верхньому ярусі два листка, на другому — чотири, на наступному — шість, і так далі, на кожному наступному ярусі на два листки більше у порівнянні з попереднім. А на самій верхівці росте ще один листочок. Хитрий Вася послав молодшу сестричку Машеньку підрахувати кількість ярусів на кожному дереві, а Вас просить написати програму, яка для кожного дерева обрахує кількість літрів води для його поливу.

Вхідні дані Кількість ярусів n ($0 \le n \le 1000$) на дереві.

Вихідні дані Вивести кількість літрів води для поливу цього дерева.

Складність: 6% — 20019/10877/8349/7881.

Справа або зліва дерева кількість листків на ярусах утворює арифметичну прогресію (2.1.3), сума якої $\frac{1+n}{2}n$. На дереві разом верхівкою $(n+1)n+1$ листків.

На **C++** (2 ms, 1.76 MiB)

```
#include <iostream>
using namespace std;
int main(){
        long n;
        cin >> n;
        cout << n*(n + 1) + 1;
}
```

На **Python** (81 ms, 8.1 MiB)

```
n=int(input())
print(n*(n+1)+1)
```

Розв'язок є і в [16].

3.6.4. e0247 Нещасливий автобус

Вітя живе досить далеко від школи, тому, щоб не запізнюватись на уроки, він їздить на автобусі. Вітя - дуже спостережливий хлопчик, він намагається помічати всі цікаві співпадання, які відбуваються в житті. Одного разу він помітив, що як тільки він сідає в автобус, у якого номер у двійковому представленні другою цифрою праворуч має одиничку, так його обов'язково викличуть до дошки відповідати заданий урок. А хто ж любить ходити до дошки?! Тим більше, якщо напередодні просидів за комп'ютером і не вивчив уроки!!! Зрозуміло, що у такому випадку загрожує "двійка"...

Допоможіть Віті скласти список автобусів, які він вважає "нещасливими"автобусами.

Вхідні дані У першому рядку задано число N ($0 \le N \le 100000$) — кількість автобусів, далі вказані номери автобусів m_i ($0 \le m_i \le 2^{31} - 1$) по одному в рядку.

Вихідні дані Виведіть кількість "нещасливих"автобусів.

Складність: 13% — 2097/986/986/859.

Для знаходження другого двійкового розряду використаємо порозрядне I (AND) з $10_2 = 2_{10}$.

На **Python** (32 ms, 5.5 MiB)

```
n,k = int(input()),0
for i in range(n): k += (int(input())&2)//2
print(k)
```

3.6.5. e9551 Сума a*a + ... + b*b

Для заданих натуральних чисел a и b знайдіть суму $a*a+...+b*b$.

Вхідні дані Два натуральних числа a і b ($1 \le a \le b \le 1000$).

Вихідні дані Виведіть значення вказаної суми.

Input	Output
3 7	135

Автор Михайло Медведєв

Складність: 9% — 248/150/148/134.

Програма на **Python** (26 ms, 5.1 MiB)

```
a,b = map(int,input().split())
s=0
for i in range(a,b+1): s += i*i
```

```
print(s)
```

3.6.6. e8609 Рекурсія – 1

Реалізуйте рекурсивну функцію:

$$f(n) = \begin{cases} 0 & \text{n=0} \\ f(n-1) & \text{n>0} \end{cases}$$

Вхідні дані Одне ціле число $n (0 \leq n \leq 1000)$.
Вихідні дані Виведіть значення $f(n)$.
Складність: 6% — 1834/1154/831/784.

Назва є оманливою, оскільки глибина рекурсії, як видно з умови задачі, може складати 999, що не допустимо при виконанні більшості коду. Можна побачити, що описана функція задає суму арифметичної прогресії (2.1.3) натуральних чисел.

Програма на **Python** (20 ms, 5.1 MiB)

```
n = int(input())
print((n+1)*n//2)
```

Програма однакова з (3.6.3).

3.6.7. e5765 Канарки

На днях в Московський зоопарк прибули нові мешканці — цілих n канарок. Поки бідні пташки нудяться в незручних тимчасових контейнерах, в залі засідань зоопарку на Раді орнітологів вирішується їх доля. А власне, вченим належить вирішити, як краще за все розподілити n канарок за наявними в зоопарку k кліткам так, щоб при цьому жодна клітка не була порожня. Оскільки головним критерієм при розміщені птахів є комфорт, орнітологів в першу чергу цікавить, скільки канарок виявиться в самій заповненої клітці (тобто в клітці з максимальним числом канарок).

Для начала, Вам, як головному (і, на жаль , єдиному) програмісту зоопарку, доручили оцінити цю величину, тобто знайти, яка мінімальна та максимальна можлива кількість птахів може виявитися в самій заповненої клітці за умови, що жодна клітина не залишиться порожньою.

Вхідні дані Два натуральних числа: кількість канарок n и кількість кліток k $(1 \leq k \leq n \leq 10^9)$.

Вихідні дані Виведіть два натуральних числа: мінімально і максимально можливу кількість канарок в самій заповненої клітці.

Складність: 7% — 7676/3301/2710/2532.

Python (101 ms, 8.1 MiB)

```
n,k = map(int,input().split())
print((n+k-1)//k, n-k+1)
```

3.6.8. e7293 Правила дорожнього руху

Степан придбав автомобіль і вирішив покатати свого молодшого брата. Петрик зайняв переднє місце пасажира. На що Степан одразу процитував правила дорожнього руху Рутенії (в країні усі дотримуються правил дорожнього руху, навіть студенти): забороняється перевозити дітей, зріст яких менший за 145 см на передньому сидінні. На скільки см слід підрости Петрику, щоб сидіти на передньому сидінні, якщо зараз його зріст складає N см.

Вхідні дані Єдиний рядок з даним числом N ($1 \leq N \leq 145$).

Вихідні дані Необхідно вивести одне число — на скільки сантиметрів слід підрости Петрику.

Ліміт часу 0.5 с

Джерело ACM-ICPC Ukraine 2014, Перший етап, 26 квітня 2014 року

Складність: 1% — 1271/938/856/844.

Java (187 ms, 24.1 MiB)

```
import java.util.Scanner;
public class e7293 {
        public static void main(String[] args) {
                Scanner in = new Scanner(System.in);
                int n = in.nextInt();
                System.out.println(145-n);
        }
}
```

Python

```
print(145-int(input()))
```

3.6.9. e0127 Бакси в банці

Папа Карло подарував Буратіно 1 долар на його перший день народження, а заощадливий Буратіно поклав подарунок у банку. Кожного наступного року папа Карло подвоював свій попередній подарунок і додавав до нього стільки доларів, скільки років виповнилось Буратіно, а той в свою чергу продовжував складати бакси у банку. На який N-й день народження у банці буде не менш ніж S доларів?

Вхідні дані Єдине число — значення S. $1 \le S \le 2^{40}$.

Вихідні дані Шукане значення N.

Автор Сергій Матвійчук

Джерело

II етап Всеукраїнської олімпіади з інформатики в Житомирській обл.

Складність: 27% — 10896/2206/2652/1941.

На **Python**

```
S,n,g,s=int(input()),0,0,0
while s<S: n+=1; g=2*g+n; s+=g
print(n)
```

3.6.10. e6059 Сума непарної послідовності

Задано непарне ціле число N, обчисліть суму всіх непарних цілих чисел між 1 і N включно.

Вхідні дані

Перший рядок вводу містить T, кількість тестових випадків. Кожен тестовий випадок містить одне ціле число N. N є від 1 до 100.

Вихідні дані

Для кожного тестового випадку виведіть значення **1+3+....+N**.

Джерело

ACM-ICPC Malaysia al-Khawarizmi Programming Contest 2011

Складність: 4% — 618/375/359/344.

Задана послідовність — арифметична прогресія (2.1.3) з $a_1 = 1$, $a_n = n$, кількістю елементів $(n+1)/2$. Її сума $S = \dfrac{1+n}{2} \cdot \dfrac{n+1}{2}$

Програма на **Python** (20 ms, 5.5 MiB)

```
for i in range(int(input())): \
    print((int(input())+1)**2//4)
```

3.6.11. e7460 Поїздка на екскурсію

Учні 10-Б класу, на осінні канікули, вирішили поїхати на екскурсію до столиці. Знаючи кількість хлопчиків n та дівчаток m, визначити скільки потрібно замовити кімнат в готелі, в якому є кімнати на k місць кожна, за умови, що хлопчиків та дівчаток поселяти разом заборонено.

Вхідні дані В одному рядку записано три числа n, m, k $(n, m, k \leq 100)$.

Вихідні дані Вивести одне число — кількість кімнат, які потрібно забронювати в готелі.

Складність: 17% — 10887/3755/3348/2765.

При діленні кількості людей на k, можемо отримати дійне число, а не ціле. Заокруглюємо число кімнат до цілого вгору.

Python (30 ms, 5.4 MiB)

```
n,m,k=map(int,input().split())
print((n+k-1)//k+(m+k-1)//k)
```

Деякі розв'язки є і в [16].

3.6.12. e6199 Дивацтва

Деякі числа непарні. Наприклад, число 3 непарне, так як не ділиться на два. Числа, які діляться на два, непарними не будуть, вони називаються парними. Більш точно, якщо число n можна представити у вигляді $n = 2 \cdot k$ для деякого цілого k, то n парне. Наприклад, $6 = 2 \cdot 3$ парне.

Деякі люди плутаються, чи є число парним або непарним. Щоб розібратися, Ви можете задати запит інтернет пошукачу "чи є число парним або непарним?"(Не виконуйте пошук! Розв'яжіть задачу!)

Вхідні дані

Починається рядком з кількістю вхідних даних n $(1 \leq n \leq 20)$. Кожен з наступних n рядків містить одне ціле число x $(-10 \leq x \leq 10)$.

Вихідні дані Для кожного x виведіть або 'x is odd', або 'x is even' в залежності від того, чи є x непарним або парним.

Джерело 2013 ACM ICPC North America - Qualification, Problem A

Складність: 5% — 4863/2514/2121/2016.

Програма на **Python** (69 ms, 8.1 MiB)

```
for i in range(int(input())):
    x=int(input())
```

```
print(x,'is odd') if x%2 else print(x,'is even')
```

3.6.13. e7330 Подільність на 3

Розглянемо послідовність **1, 12, 123, 1234, 12345, 123456, 1234567, 12345678, 123456789, 12345678910, 1234567891011, ...**.

Скільки елементів даної послідовності серед перших n ділиться на три.

Вхідні дані Одне натуральне число n ($1 \le n \le 2^{31} - 1$).

Вихідні дані Вивести одне знайдене число.

Джерело 2014 ACM-ICPC Україна, 2-й тур Вересень 13, Задача G

Складність: 20% — 3880/1172/1172/943.

Очевидно, що перебрати $2^{31} - 1 = 2\,147\,483\,647$ елементів послідовності за вказаний час неможливо.

На **Python** (31 ms, 5.4 MiB)

```
n=int(input())
print(n//3*2+(n%3==2))
```

Практично такий же розв'язок задачі і в [16].

3.6.14. e4743 Подорож Нільса з дикими напівгусками

Коли Нільс подорожував з дикими гусками, його зграя пролітала над озерами. Було вирішено зробити перерву, і гуски почали сідати на озера. На перше озеро сіла половина усієї зграї та ще півгуски. На друге озеро сіла половина зграї, що залишилась, та ще півгуски. І так далі, доки усі гуски не розсілись на K озерах.

Тепер Нільс хоче визначити, скільки гусок було у зграї спочатку. Допоможіть йому це зробити!

Вхідні дані У вхідному файлі записано єдине число K ($1 \le K \le 20$).

Вихідні дані

У вихідний файл виведіть початкову кількість гусок у зграї.

Input	**Output**
1	1

Складність: 15% — 1027/498/514/439.

Нехай n — початкова кількість гусок в зграї. На першому озері $\left(\frac{n}{2} + \frac{1}{2}\right)$ гусок сіли на озеро, $\left(\frac{n}{2} - \frac{1}{2}\right)$ — залишились в зграї.

Позначимо $m = (n+1)/2$. Тоді залишились в зграї — $m-1$, m — вибули (сіли на озеро), $n = 2m-1$. На наступному озері залишок в зграї можна аналогічно надати в вигляді $m-1 = 2i-1$, звідки $m = 2i$. Отже на кожному озері вибуває вдвічі менше гусок ніж на попередньому ($m = 2^j$). Враховуючи, що на останньому озері лишається 0 та вибуває 1, маємо $n = 2m-1 = 2^k - 1$.

Програма на **Python** (33 ms, 5.5 MiB)

```
print(2**int(input())-1)
```

3.6.15. e6777 Автобус

Автобус з n пасажирами відкриває двері на автобусній зупинці. Рівно половина пасажирів плюс півпасажира виходить. На наступній зупинці знову виходить з автобуса половина пасажирів плюс півпасажира. Так продовжується k зупинок. Знаючи, що на останній зупинці автобус став пустим, і ніхто не постраждав під час поїздки, визначте початкову кількість людей n в автобусі.

Вхідні дані Перший рядок містить кількість тестів t. Кожен тест містить в окремому рядку кількість зупинок k ($1 \le k \le 30$).

Вихідні дані Для кожного тесту вивести в окремому рядку початкову кількість пасажирів.

Input	**Output**
2	1
1	7
3	

Джерело 2013 ACM Central Europe, November 15-17, Problem L
Складність: 7% — 1996/1072/914/850.

Використовуємо, спираємось на 3.6.14.

Програма на **Python** (43 ms, 5.5 MiB)

```
for i in range(int(input())): print(2**int(input())-1)
```

3.6.16. e2806 Числа

Задано натуральне число N.

Напишіть програму, яка знаходить кількість натуральних чисел, що не перевищують N і не діляться на жодне з чисел 2, 3, 5.

Вхідні дані Один рядок, у якому міститься число N ($1 \le N \le 1\,000\,000\,000$).

Вихідні дані Вивести знайдене число.

Автор Анатолій Присяжнюк

Джерело

II етап Всеукраїнської олімпіади школярів 2012-2013, м. Бердичів

Складність: 39% — 3253/633/867/529.

Маємо справу з перетином множин чисел, які $\vdots 2$, $\vdots 3$, $\vdots 5$.

Python (27 ms, 7.7 MiB)

```
n=int(input())
print(n-n//2-n//3-n//5+n//6+n//10+n//15-n//30)
```

Розв'язок є і в [16].

3.6.17. e2817 Двійкові числа

Для заданого додатного цілого числа n вивести позиції усіх 1 у його двійковому поданні. Позиція молодшого біта має номер 0.

Позиції 1 у двійковому поданні числа 13 — це 0, 2, 3.

Напишіть програму, яка для кожного набору даних:

. читає натуральне число n,

. обчислює позиції 1 у двійковому поданні n,

. виводить результат.

Вхідні дані У першому рядку вхідного файлу міститься одне натуральне число d, яке вказує кількість наборів вхідних даних, $1 \le d \le 10$. Вхідні дані задано нижче.

Кожен набір даних складається рівно з одного рядка, який містить рівно одне ціле число n, $0 \le n \le 10^6$.

Вихідні дані Вихід повинен складатись рівно з d рядків — по одному рядку для кожного набору вхідних даних.

Рядок i, $1 \le i \le d$, повинен містити зростаючу послідовність цілих чисел, відокремлених одним пропуском — позиції 1 у двійковому поданні i-го числа, отриманого з вхідних даних.

Джерело

II етап Всеукраїнської олімпіади школярів 2012-2013, м. Бердичів

Складність: 16 % — 1283/512/524/439.

Python (27 ms, 5.5 MiB)

```
d=int(input())
for i in range(d):
    n,b,k,f=int(input()),1,0,0
    for j in range(20):
        if n&b>0:
            if f: print(' ',j,end='')
            else: print(j,end=''); f=1
        b*=2
    print()
```

3.6.18. e3254 01110001, ось запитання

Як відомо, числа в двійковій системі записують за допомогою цифр 0 та 1. Ваше завдання —- перевести число з двійкового подання в десяткове.

Вхідні дані Двійковий запис цілого невід'ємного числа. У записі числа не більше 15 цифр. Запис може починатися з нулів.

Вихідні дані Вивести десятковий запис вхідного двійкового числа.

Джерело The 2012 All-Ukrainian Collegiate Programming Contest Round I Training Contest 19 April 2012

Складність: 7% — 2842/1647/1411/1306.

Python

```
print(int(input(),2))
```

Розв'язок є і в [16].

3.6.19. e0318 Біноміальні коефіцієнти 1

Нехай n — ціле невід'ємне число. Позначимо $n! = 1 \times 2 \times ... \times n$ $(0! = 1)$ та

$$C_n^k = \frac{n!}{k!(n-k)!} \ (0 \le k \le n)$$

За заданими n та k обчислити $C(n,k)$.

Вхідні дані Перший рядок містить кількість тестів t $(t \le 50)$. Кожний з наступних t рядків містить два цілі числа n та k $(0 \le n < 2^{64}$, $0 \le C(n,k) < 2^{64})$.

Вихідні дані Вивести t рядків, кожен з яких містить значення $C(n,k)$ для відповідного тесту.

Складність: 46% — 5729/682/919/496.

Вважаючи на можливі великі значення n та дуже швидке зростання $n!$, задачу неможливо розв'язувати безпосереднім знаходженням факторіалу (без значних обчислювальних втрат). Враховуючи знаменник, необхідно мінімізувати кількість множень та ділень. Найпростіше маємо розрахунок добутку $n \times (n-1) \times ... \times (n-k+1)$ та $2 \times 3 \times ... \times k$ (де k — найменше число в знаменнику), чим і скористуємось. Реально добуток чисельника поділимо на множники вказаного знаменника.

В оптимальному варіанті, знаючи що результат ціле число, підбираємо множники з чисельника та знаменника, так щоб результат ділення був цілим і попадав в розрядну сітку процесора (64 біти), як це гарантується в умові задачі.

Python (29 ms, 5.4 MiB)

```
for i in range(int(input())):
    n,k = map(int,input().split())
    k,r = max(k,n-k),1
    for i in range(k+1,n+1): r*=i
    for i in range(2,n-k+1): r//=i
    print(r)
```

3.6.20. e4887 Цифри

Назвемо сумою цифр числової послідовності суму цифр усіх її чисел. Наприклад, для послідовності чисел 14, 22, 239 сума цифр буде рівною $(1+4)+(2+2)+(2+3+9)=23$.

Ваша задача — для заданого n знайти суму цифр наступної числової послідовності: $1, 2, 3, ..., 10^n - 1$

Вхідні дані У першому рядку вхідного файлу знаходиться ціле число n ($1 \le n \le 100\,000$).

Вихідні дані Виведіть у вихідний файл одне число — шукану суму цифр числової послідовності.

Джерело Blitz Contest by SPbETU & Michael Dvorkin, Petrozavodsk Winter Training Session, January 31, 2006

Складність: 28% — 263/75/88/63.

Python (25 ms, 7.8 MiB)

```
n=int(input())
print(45*n,'0'*(n-1),sep='')
```

3.6.21. e2710 Трикутник Паскаля

Трикутник Паскаля — це числовий трикутник, по краям якого стоять одиниці, а кожне число всередині дорівнює сумі двох чисел вгору-праворуч і вгору-ліворуч.

Із-за помилки набірника трикутник Паскаля виявився записаним у рядок і утворилась послідовність виду 1, 1, 1, 1, 2, 1, 1, 3, 3, 1, 1, 4, 6, 4, 1, ...

Вхідні дані Задано один рядок, який містить натуральне число N ($N \leq 600$).

Вихідні дані Потрібно вивести один рядок, який містить N-ий член утвореної послідовності.

Складність: 16% — 564/238/244/204.

Використовуємо безпосередній розрахунок біноміальних коефіцієнтів (за потреби можна скористатись 3.6.20).

Python (32 ms, 5.5 MiB)

```
from math import ceil,factorial
N = int(input())
n = ceil(((1+8*N)**.5-1)/2)-1
k = N-n*(n+1)//2-1
print(factorial(n)//factorial(k)//factorial(n-k))
```

3.6.22. e7327 Сходові числа

Розглянемо числа вигляду $a\hat{}(a\hat{}(a\hat{}...)$, де a — натуральне число, яке в записі зустрічається два і більше разів, $\hat{}$ — операція піднесення до степеня. Назвемо такі числа "Сходовими"(число + сходи). Наприклад $27 = 3\hat{}3$ і $16 = 2\hat{}(2\hat{}2)$ є сходовими числами. Число 1 є також сходовим числом, так як $1 = 1\hat{}1$. А числа 2, 3, 5 не є сходовими числами, бо їх не можна подати у потрібному вигляді. Знайдіть кількість сходових чисел на проміжку від 1 до n включно.

Вхідні дані Одне число n ($1 \leq n \leq 10^9$).

Вихідні дані

Вивести кількість сходових чисел, які не перевищують n.

Джерело

2014 ACM-ICPC Україна, 2-й Раунд, Вересень 13, Задача D

Складність: 20% — 1659/349/364/292.

Сходові числа ростуть дуже швидко з номером, а отримуються довго. Тому розраховуємо ці числа, заносимо в масив, а наша програма тільки використовує їх.

На **Python** (32 ms, 7.7 MiB)

```
n,p,k=int(input()),[1,4,16,27,256,3125,46656,65536, \
      823543,16777216,387420489,1000000001],0
while p[k]<=n: k += 1
print(k)
```

3.6.23. e9636 Діно та два кольори

Діно ненавмисно на стіні школи намалював червону смужку довжиною a. До кінця перерви це потрібно приховати. Діно згадав, що у нього є дві банки з білою фарбою, яких вистачає, щоб намалювати білу смужку довжиною b. Він прагне як найшвидше червону смужку замалювати білою, використавши наявну в нього білу фарбу. Зрозуміло, що червону смужку він може замалювати повністю, або частково, але він хоче, щоб червоної смужки залишилось як найменше. Допоможіть Діно це зробити.

Вхідні дані Два цілих числа a і b ($1 \le a, b \le 10^9$).

Вихідні дані Виведіть можливу мінімальну довжину червоної смужки, що залишиться на стіні, або виведіть 0, якщо Діно її замалює повністю.

Складність: 15% — 597/262/282/240.

На **Python** (22 ms, 5.1 MiB)

```
a,b=map(int,input().split())
print((a-2*b)*(a>2*b))
```

3.6.24. e6388 Муха Фон Неймана

Наступна задача була запропонована Джону Фон Нейману:

Два велосипедисти a і b починають поїздку назустріч один одному в один і той же час з місць, що знаходяться на відстані 250 один від одного, a рухається зі швидкістю 10 миль на годину, b рухається зі швидкістю 15 миль на годину. В цей же час муха злітає з колеса велосипедиста a і рухається назустріч до b, потім розвертається і летить назад. Поки велосипедисти наближаються одна до одної, муха продовжує літати між

ними, торкаючись кожного разу переднього колеса велосипедистів, поки, нарешті, не буде розчавлена колесами зустрілися велосипедів. Так як муха літає швидше кожного з велосипедистів, вона робить нескінченну кількість польотів, при цьому пройшовши кінцеве відстань (нескінченний ряд сходиться). Яка відстань пролетіла муха?

Фон Нейман миттєво обрахував нескінченний ряд (порахував про себе!), і отримав вірну відповідь: 200 миль.

Вам належить написати програму, яка розв'язує більш загальну задачу, с різними початковими відстанями та швидкостями.

Вхідні дані Перший рядок містить кількість тестів $p (1 \le p \le 1000)$.

Кожен тест складається з одного рядка, містить п'ять чисел: номер тесту n і чотири дійсних числа: початкові відстані між велосипедистами d $(10 \le d \le 1000)$, швидкість первого велосипедиста a $(1 \le a \le 30)$ в милях на годину, швидкість другого велосипедиста b $(1 \le b \le 30)$ в милях і швидкість мухи f $(a \le b \le f \le 50)$ в милях на годину.

Вихідні дані Для кожного тесту вивести в окремому рядку номер тесту, пробіл, і кількість миль, які пролетіла муха (нескінченна сума відстаней, описаних в умові) з точністю до двох десяткових знаків.

Джерело

2013 ACM Greater New York Region, Жовтень 27, Задача B

Складність: 8% — 948/463/419/385.

Легендарна задача фон Неймана [32]. Час руху велосипедистів до зустрічі або час польоту мухи — $d/(a+b)$. За цей час муха пролетить $d/(a+b) \times f$.

Програма на **Pascal** (2 ms, 0.65 MiB)

```
var p,n,i:integer;
    d,a,b,f:real;
begin
  readln(p);
  for i:=1 to p do
    begin
      readln(n,d,a,b,f);
      writeln(n,' ',d/(a+b)*f:0:2)
    end
end.
```

Також на C++ [25]

```
#include <iostream>
```

```
using namespace std;
int main() {
  double p, d, a, b, f, t, flyDist;
  int n;
  cin >> p;
  for (int i = 0; i < p; i++){
    cin >> n >> d >> a >> b >> f;
    t = d / (a + b);
    flyDist = f * t;
    cout.precision(2);
    cout << fixed << n << "␣" << flyDist << endl;
  }
}
```

3.6.25. e0036 Змій Горинич

В деякому царстві жив Змій Горинич. У нього було N голів та M хвостів. Іван-царевич вирішив знищити губителя людських душ, для чого йому його кума Баба Яга подарувала чарівний меч, оскільки тільки ним можна вбити Змія Горинича. Якщо відрубати одну голову, то на її місці виростає нова, якщо відрубати хвіст, то замість нього виросте 2 хвости. Якщо відрубати два хвости, то виросте 1 голова, і тільки коли зрубати 2 голови, то не виростає нічого. Змій Горинич гине тільки в тому випадку, коли йому відрубати всі голови і всі хвости. Визначити мінімальну кількість ударів мечем, потрібну для знищення Змія Горинича.

Вхідні дані Два числа N, M $(0 \leq N, M \leq 1000)$.

Вихідні дані Єдине число — мінімальна кількість ударів мечем, або -1, якщо знищити Змія Горинича неможливо.

Автор Анатолій Присяжнюк

Складність: 30% — 4538/1164/1523/1064.

На **Pascal**

```
var n,m,k:word;
begin
  read(n,m);
  if (m=0) and (n mod 2=1)then begin writeln(-1); \
    exit end;
  k:=(m+1)div 2;
  n:=n+k;
  writeln(k+m mod 2 +(n+1)div 2 + 3*(n mod 2))
```

```
end.
```

3.6.26. e4739 Решето Ератосфена

За заданими числами a та b вивести усі прості числа з інтервалу від a до b включно.

Вхідні дані Два числа a та b ($1 \le a \le b \le 100000$).

Вихідні дані Ввести в одному рядку усі прості числа з інтервалу від a до b включно.

Складність: 24% — 5216/1338/1296/980.

На **Pascal** (4 ms, 0.68 MiB)

```
var p:array[1..100000]of boolean;
    a,b,i,k:longint;
    f:boolean;
begin
  read(a,b);
  p[1]:=true;
  for i:=2 to round(sqrt(b)) do
  begin
    if p[i] then continue;
    k:=i*i;
    while k<=b do begin p[k]:=true; k:=k+i end;
  end;
  for i:=a to b do
    if not p[i] then
      if f then write(' ',i) else begin write(i);
        f:=true end;
  writeln
end.
```

На **C++** (8 ms, 0.68 MiB)

```
#include <iostream>
using namespace std;
int main() {
  int a, b, i;
  cin >> a >> b;
  bool *arr = new bool[b + 1];
  for (i = 2; i <= b; i++) arr[i] = true;
  arr[1] = false;
```

```
  for (i = 2; i*i <= b; i++)
    if (arr[i])
      for (int j=i*i; j<= b; j+=i) arr[j]=false;
  for (i = a; i <= b; i++)
    if (arr[i]) cout << i << "␣";
}
```

На **Python** (79 ms, 8.5 MiB)

```
a,b=map(int,input().split())
p=[True for i in range(b+1)]
p[0]=p[1]=False
for i in range(2,b):
    if p[i]:
        for j in range(i**2,b+1,i): p[j]=False
for i in range(a,b+1):
    if p[i]: print(i,end='␣')
```

3.6.27. e0571 Найбільший спільний дільник

Задано n натуральних чисел. Напишіть програму, яка обчислює найбільший спільний дільник цих чисел.

Вхідні дані У першому рядку знаходиться натуральне число n ($n \leq 1000$) — кількість чисел. Далі йде n натуральних чисел, кожне з яких не перевищує $2 \cdot 10^9$.

Вихідні дані Виведіть єдине число — найбільший спільний дільник заданих чисел.

Джерело ЛКШ 2009

Складність: 12% — 5045/2338/2207/1953.

Використовуємо алгоритм Евкліда.

На **Python** (19 ms, 5.2 MiB)

```
def gcd(n,m):
    while n*m: n%=m; n,m=m,n
    return n+m
input()
d=list(map(int,input().split()))
g=d[0]
for i in d: g=gcd(g,i)
print(g)
```

3.7. Комбінаторика

3.7.1. e1288 n-значні числа

Скільки натуральних n-значних чисел починається з цифри a або цифри b?

Вхідні дані

Три цілих числа: n ($0 < n \leq 10^6$) , a та b. Всі дані, як і сама умова задачі, задані у десятковій системі числення.

Вихідні дані

Вивести кількість натуральних n-значних чисел, що починаються з цифри a або цифри b.

Input	**Output**
3 3 4	200

Складність: 45% — 4958/742/922/504.

Python (30 ms, 7.2 MiB)

```
n,a,b=map(int,input().split())
if a == b == 0: print(0)
else:
    r='1' if a == b or a*b == 0 else '2'
    print(r+'0'*(n-1))
```

3.7.2. e1355 Кількість N-значних чисел, що містить цифру 7

Знайти K — кількість N-значних натуральних чисел, що мають у своєму запису хоча б одну цифру 7.

Вхідні дані Одне натуральне число N ($1 \leq N \leq 10$).

Вихідні дані Шукане число K.

Складність: 32% — 1996/513/614/415.

Найкраще скористуємось [16]. Для n-значних чисел є 9 варіантів першої цифри та 10 всіх інших, тобто всього є $9 \times 10^{n-1}$ таких чисел. Для n-значних чисел, що не містять жодної цифри 7, є 8 варіантів першої цифри та 9 всіх інших, тобто всього є $8 \times 9^{n-1}$ таких чисел.

Модифікований розв'язок з [16] на **Python**

```
m = int(input())-1
print(9*10**m-8*9**m)
```

3.7.3. e2385 Кількість перестановок

За заданим натуральним числом n знайти кількість різних перестановок чисел від 1 до n.

Вхідні дані Одне число n ($1 \le n \le 12$).

Вихідні дані Вивести кількість різних перестановок чисел від 1 до n.

Складність: 5% — 2582/1714/1506/1433.

Програма на **Python** (41 ms, 5.5 MiB)

```
from math import factorial
print(factorial(int(input())))
```

Розв'язок такий же як (3.4.49) та (3.4.49).

3.7.4. e1290 Номерний знак

Міжнародний номерний реєстраційний знак легкового автомобіля складається з **A** арабських цифр і **B** великих літер латинського алфавіту. Будемо вважати, що для забезпечення унікальності номера дозволено використовувати довільну послідовність літер і цифр.

Скільки існує різних таких номерів?

Вхідні дані У єдиному рядку через пропуск задано 2 невід'ємних цілих числа **B** та **A**. Обидва числа не перевищують 26.

Вихідні дані Єдине число — відповідь до задачі.

Input	**Output**
3 3	17576000

Складність: 39% — 2083/402/566/347.

Перевіряємо приклад та робимо висновок, що ми маємо справу не з числами в номерному знаці, тому він може починатись і з нуля. Таким чином для кожного розряду 26 варіантів для літер та 10 — для цифр.

Програма на **Python** (23 ms, 775 MiB)

```
b,a=map(int,input().split())
print(26**b,'0'*a,sep='')
```

3.7.5. e1289 Ланч

Влад бажає взяти з собою для ланчу пару фруктів. У нього є **a** різних бананів, **b** різних яблук та **c** різних груш. Скількома способами він може обрати 2 різні фрукти з того що у нього є?

Вхідні дані В одному рядку задано три невід'ємні числа: **a**, **b**, **c**. Усі числа не перевищують 10^6.

Вихідні дані Вивести кількість способів, якими можна обрати 2 фрукти різного виду.

Складність: 12% — 7515/3436/3110/2746.

Pascal

```
var a,b,c:int64;
begin
   read(a,b,c);
   writeln(a*b+b*c+a*c)
end.
```

C++

```
#include <iostream>
using namespace std;
int main() {
    long int a,b,c;
    cin >> a >> b >> c;
    cout << a*b+b*c+a*c;
}
```

3.7.6. e0390 Анаграми

Анаграмою слова називається довільна перестановка всіх літер слова. Наприклад, зі слова **SOLO** можна отримати 12 анаграм: **SOLO, LOSO, OSLO, OLSO, OSOL, OLOS, SLOO, LSOO, OOLS, OOSL, LOOS, SOOL**.

Напишіть програму, яка виводить кількість різних анаграм, які можна отримати з цього слова.

Вхідні дані Слово, кількість літер в якому не перевищує 14.

Вихідні дані Кількість різних анаграм.

Складність: 30% — 4634/1224/1366/958.

На **Python** (36 ms, 5.5 MiB)

```
import math
r=input()
k,s=math.factorial(len(r)),set(r)
for i in s: k//=math.factorial(r.count(i))
```

```
print(k)
```

3.7.7. e1287 Змагання з тенісу

Необхідно сформувати команду, яка буде представляти навчальний заклад у змаганнях з тенісу. У секції тенісу займається A дівчат і B хлопців. Скільки різних змішаних пар можна вибрати для участі у змаганнях?

Вхідні дані У єдиному рядку через пропуск знаходиться 2 цілих невід'ємних числа A та B, які не перевищують 10^6.

Вихідні дані Єдине число — відповідь до задачі.

Ліміт часу 0.1 с

Складність: 14% — 5123/2172/2227/1906.

На **C#**

```
using System;
namespace e1287 {
  class Program {
    static void Main(string[] args) {
      string[] r = Console.ReadLine().Split();
      Console.WriteLine(Convert.ToInt64(r[0]) *
        Convert.ToInt64(r[1]));
    }
  }
}
```

На **Python** (72 ms, 8.1 MiB)

```
a,b=map(int,input().split())
print(a*b)
```

3.7.8. e1326 У хокей грають справжні...

Лісові жителі вирішили провести хокейний турнір між N командами. Скількома способами можуть бути розподілені комплекти золотих, срібних та бронзових медалей, якщо одне призове місце може зайняти лише одна команда?

Вхідні дані У єдиному рядку розміщено єдине натуральне число N, яке не перевищує 100.

Вихідні дані Єдине число — шукана кількість способів.

Ліміт часу 0.18 с
Автор Анатолій Присяжнюк
Складність: 20% — 6177/1671/1810/1441.

На **Python** (29 ms, 5.5 MiB)

```
from math import factorial
n=int(input())
print(factorial(n)//factorial(n-3)) if n>2 else print(n)
```

3.7.9. e1327 Тури на шаховій дошці

Ще у дитинстві маленького Гаріка зацікавило питання: а скількома способами на шаховій дошці розміром $n \times n$ можна розставити n тур так, щоб вони не били одна одну. Він дуже довго розв'язував цю задачку для кожного варіанту, а коли розв'язав — закинув шахи.

А як швидко Ви справитесь з цією задачкою?

Вхідні дані Розмір шахової дошки — натуральне число, яке не перевищує 1000.

Вихідні дані Виведіть відповідь, знайдену Гаріком.

Автор Анатолій Присяжнюк
Складність: 27% — 3319/1025/1084/789.

На **Python** (36 ms, 7.7 MiB)

```
from math import factorial
print(factorial(int(input())))
```

Код програми збігається з 3.4.49 та (3.4.49).

3.7.10. e1328 Малюнки на аркуші в клітинку, e7341 Кількість прямокутників

У Василька є аркуш в клітинку, який містить N клітинок по горизонталі та M клітинок по вертикалі, причому лінії клітинок аркуша на краю також видно. Скільки різних прямокутників може на цьому аркуші намалювати Василько, якщо малювати він вміє лише по лініям?

Вхідні дані У єдиному рядку через пропуск знаходяться два числа M та N. Всі вхідні дані натуральні числа, які не перевищують 10^{10000}.

Вихідні дані Єдине число — шукана кількість прямокутників.

Автор Анатолій Присяжнюк
Складність: 41% — 1000/254/345/202.

Вздовж однієї сторони — 1 прямокутник розміром n клітинок, 2 розміром $n-1$, ..., n клітинок розміром 1. Всього — сума арифметичної прогресії натуральних чисел.

На **Python** (55 ms, 7.7 MiB)

```
n,m=map(int,input().split())
print(n*(n+1)*m*(m+1)//4)
```

e7341 Кількість прямокутників Не вельми завзяті учні на уроці інформатики вигадали собі гру. На аркуші в клітинку малювався прямокутник довільного розміру у такий спосіб, що сторони лежали на сторонах клітинки. Необхідно було порахувати кількість всіх можливих прямокутників, що можна утворити використовуючи сторони клітинок. Вчителю інформатики, Альберту Верверовучу, все це надоїло і він заставив цих учнів написати програму, яка буде робити це обчислення автоматично.

Вихідні дані Два натуральних числа через пропуск які не перевищують 10000 — розміри прямокутника.

Одне число -кількість шуканих прямокутників.

Складність: 39% — 1182/337/405/249.

3.8. Дійсні числа

3.8.1. e8876 Ціле число

Задано дійсне число **n**. Вивести **Ok**, якщо число **n** ціле та **No** у протилежному випадку.

Вхідні дані Одне дійсне число **n**.

Вихідні дані Вивести **Ok**, якщо число **n** ціле та **No** інакше.

Автор Матвійчук Сергій Володимирович

Джерело Серія задач "Абетка програмування"

Складність: 11% — 2303/938/1290/948.

Перетворимо введене дійсне у ціле число і порівняємо їх між собою.

На **Python** (24 ms, 5.1 MiB)

```
n = float(input())
print('Ok') if n == int(n) else print('No')
```

3.8.2. e7829 Сума елементів

Дано послідовність з n дійсних чисел. Знайти суму всіх її елементів.

Вхідні дані

В першому рядку записано число n ($n \leq 100$). В наступному рядку записано n дійсних чисел, кожне з яких не перевищує за модулем 100.

Вихідні дані Вивести суму всіх елементів послідовності.

Складність: 6% — 10529/5534/3511/3292.

Враховуємо, спираємось на

Python (23 ms, 7.5 MiB)

```
input()
print(sum(map(float,input().split())))
```

Розв'язок задачі на Python також можна знайти в [16] (стор.60).

C++.

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    int n;
    float s=0, a;
    cin >> n;
    for(int i=0; i<n; i++){cin >> a; s += a;}
    cout << s << endl;
}
```

Розв'язок задачі на C++ також можна знайти в [16] (стор.165). Використовуються масиви.

3.8.3. e0957 Квадратний корінь

Знайти квадратний корінь суми цифр трицифрового натурального числа.

Вхідні дані Одне натуральне трицифрове число.

Вихідні дані Вивести квадратний корінь суми цифр з 3 десятковими цифрами.

Складність: 4% — 6936/3994/3523/3381.

Використовуємо, спираємось на **??**

C#

```
using System;
class Program {
    static void Main(string[] args) {
        string n = Console.ReadLine();
        Console.WriteLine("{0:0.000}",Math.Sqrt(n[0]+
          n[1]+n[2]-3*'0'));
    }
}
```

Python

```
import math
n,s = input(),0
for i in n: s += int(i)
print("%.3f"%math.sqrt(s))
```

Також враховуючи 3.4.14 **Python** (35 ms, 5 MiB)

```
print(sum(map(int,list(input())))**.5)
```

3.8.4. e0910 Середнє арифметичне додатних

Задано послідовність дійсних чисел. Визначити середнє арифметичне додатних чисел.

Вхідні дані У першому рядку задано кількість чисел n ($0 < n \leq 100$). У наступному рядку задано n дійсних чисел, значення яких не перевищують за модулем 100.

Вихідні дані Вивести середнє арифметичне додатних чисел з 2 десятковими знаками. У випадку відсутності додатних чисел вивести повідомлення "Not Found"(без лапок).

Джерело ДПА-2010 Варіант 10

Складність: 12% — 25798/8521/7256/6354.

На **Pascal** (20 ms, 5.1 MiB)

```
var h,i,k:byte;
    a,s:real;
begin
  read(h);
  for i:=1 to h do
    begin
```

```
        read(a);
        if a<=0 then continue;
        s:=s+a;
        inc(k)
      end;
   if k=0 then writeln('Not Found')
          else writeln(s/k:0:2)
end.
```

3.8.5. e0927 Кількість іграшок

Задано кількість видів іграшок в магазині, кількість іграшок кожного виду та вартість іграшки кожного виду. Визначити кількість іграшок, вартість яких менша за 50 грн. **Вхідні дані**

У першому рядку задано кількість наявних у прейскуранті видів іграшок n ($0 \le n \le 1000$). У наступних n рядках задано по 2 числа через пропуск: спочатку кількість іграшок a ($0 \le a \le 1000$) чергового виду та їх ціна ($0 < b \le 10000$) в грн.

Вихідні дані Вивести кількість іграшок, вартість яких менша за 50 грн.

Джерело ДПА-2010 Варіант 27

Складність: 9% — 13164/6059/5279/4792.

На **Python**

```
k = 0
for i in range(int(input())):
    a,b = input().split()
    if float(b)<50: k += int(a)
print(k)
```

3.8.6. e0931 Відношення добутку до суми

Обчислити відношення добутку цифр натурального числа до їх суми.

Вхідні дані Натуральне число n, що не перевищує $2 \cdot 10^9$.

Вихідні дані Вивести відношення добутку цифр числа n до їх суми з 3 десятковими цифрами.

Складність: 16% — 18930/6292/5643/4728.

На **Python**

```
n,p,s = input(),1,0
for i in n: d = int(i); p *= d; s += d
print('%.3f'%(p/s))
```

На **Pascal** (21 ms, 7,5 MiB)

```
var n:string;
    m,d,s,i:longint;
begin
      readln(n);
      m:=1;s:=0;
      for i:=1 to length(n) do
          begin
              d:=ord(n[i])-ord('0');
              m:=m*d;
              s:=s+d;
          end;
      writeln(m/s:0:3);
end.
```

3.8.7. e8239 Функція — 1

Реалізуйте функцію $f(x) = x^3 + 2 \cdot x^2 - 3$.

Вхідні дані Кожний рядок містить одне дійсне число x.

Вихідні дані Для кожного значення x вивести в окремому рядку значення функції $f(x)$ з чотирма десятковими знаками.

Автор Михайло Медведєв

Джерело Мова програмування C

Складність: 15% — 4014/1491/1343/1139.

Програма на **C++** (2 ms, 1.7 MiB)

```
#include <iostream>
#include <cmath>
using namespace std;
int main(){
    double x;
    while (cin >> x)
        printf("%.4f\n", pow(x,3)+2*pow(x,2)-3);
}
```

Програма на **Python** (20 ms, 5.1 MiB)

```
for line in open('input.txt'):
    x = float(line)
    print(x**3+2*x**2-3)
```

3.8.8. e8240 Функція — 2

Реалізуйте функцію $f(x) = sqrt(x) + 2 * x + sin(x)$, де $sqrt$ — функція квадратного кореня.

Вхідні дані Кожний рядок містить одне дійсне число x.

Вихідні дані Для кожного значення x вивести в окремому рядку значення функції $f(x)$ з чотирма десятковими знаками.

Автор Михайло Медведєв

Джерело Мова програмування C

Складність: 7% — 2090/1196/1036/960.

Програма на **C++** (2 ms, 1.7 MiB)

```
#include <iostream>
#include <cmath>
using namespace std;
int main() {
    double x;
    while (cin >> x)
        printf("%.4f\n", sqrt(x) + 2 * x +sin(x));
}
```

3.8.9. e8241 Функція — 3

Реалізуйте функцію $f(x, y) = x^2 + sin(x * y) - y^2$.

Вхідні дані Кожний рядок містить два дійсних числа x та y.

Вихідні дані Для кожного тесту вивести в окремому рядку значення функції $f(x)$ з чотирма десятковими знаками.

Автор Михайло Медведєв

Джерело Мова програмування C

Складність: 7% — 2363/1208/1007/934.

Програма на **C++** (2 ms, 1.7 MiB)

```
#include <iostream>
#include <cmath>
```

```
using namespace std;
int main() {
    double x,y;
    while (cin >> x >> y)
        printf("%.4f\n",pow(x,2)+sin(x*y)-pow(y, 2));
}
```

3.8.10. e0112 Торт

На день народження спадкоємця Тутті королівський кухар приготував великий святковий торт, який було подано на стіл Трьом Товстунам. Перший товстун сам міг би повністю його з'їсти за t_1 годин, другий — за t_2 годин, а третій — за t_3 годин.

Скільки часу потрібно товстунам, щоб з'їсти увесь святковий торт разом?

Вхідні дані Єдиний рядок містить три не від'ємні цілі числа t_1, t_2 та t_3, кожне з яких не перевищує 10000.

Вихідні дані Вивести час в годинах, за який товстуни одночасно можуть з'їсти торт. Результат округлити до двох десяткових знаків.

Складність: 21% — 15276/4134/4375/3445.

Швидкість поїдання торту $v = 1/t$. При сумісному поїданні швидкості складаються. Виняток — хоча б один Товстун з'їдає торт миттєво ($t = 0,\ v = \infty$).

Програма на **Pascal**

```
var t,t2,t3:integer;
begin
  read(t,t2,t3);
  if t*t2*t3=0 then writeln('0.00')
  else writeln(1/(1/t+1/t2+1/t3):0:2)
end.
```

Програма на **Python**

```
t,T,t3=map(int, input().split())
print('%.2f'%(1/(1/t+1/T+1/t3))) if t*T*t3 else \
    print('0.00')
```

3.9. Комплексні числа

3.9.1. e9531 Комплексні числа: додавання та віднімання

Дано два комплексних чисел. Найдіть їх суму або різницю.

Вхідні дані В кожному рядку задано приклад на додавання або віднімання комплексних чисел. Комплексне число задається в форматі $a + bi$ або $a - bi$, де a ціле, b ціле невід'ємне. Дійсна та уявна частина кожного комплексного числа по модулю не перевищує 10^9.

Вихідні дані Для кожного вхідного прикладу виведіть відповідь в окремому рядку.

Input	Output
2+3i + 7-4i	9-1i
12-4i - 5-4i	7+0i
-1-1i - -1-1i	0+0i
5-2i - -7+12i	12-14i

Автор Михайло Медведєв

Складність: 29% — 84/34/35/25.

Використаємо вбудовану роботу з комплексними числами.

Програма на **Python** (26 ms, 5.1 MiB)

```
for line in open('input.txt'):
    Z,s,z = line.split()
    Z = complex(Z.replace('i','j'))
    z = complex(z.replace('i','j'))
    r = Z+z if s == '+' else Z-z
    s = '' if r.imag < 0 else '+'
    print("%d%s%d%s"% (r.real,s,r.imag,'i'))
```

3.9.2. e9532 Комплексні числа: множення та ділення

Дано два комплексних чисел. Найдіть їх добуток або частку.

Вхідні дані В кожному рядку задано приклад на множення або ділення комплексних чисел. Комплексне число задається в форматі $a+bi$ або $a - bi$, де a ціле, b ціле невід'ємне. Дійсна та уявна частина кожного комплексного числа по модулю не перевищує 10^9.

Вихідні дані Для кожного вхідного прикладу виведіть відповідь в окремому рядку.

Input	Output

2+3i * 7-4i	26.00+13.00i
12-4i / 5-4i	1.85+0.68i
-1-1i * -1-1i	0.00+2.00i
5-2i / -7+12i	-0.31-0.24i

Автор Михайло Медведєв
Складність: 50% — 2/1/2/1.

Використаємо вбудовану роботу з комплексними числами.
Програма на **Python** (21 ms, 5.1 MiB)

```
for line in open('input.txt'):
    Z,s,z = line.split()
    Z = complex(Z.replace('i','j'))
    z = complex(z.replace('i','j'))
    r = Z*z if s == '*' else Z/z
    s = '' if r.imag < 0 else '+'
    print("%.2f%s%.2f%s"% (r.real,s,r.imag,'i'))
```

3.10. Обробка рядків

3.10.1. e6592 Прекрасний Єкатеринбург

Єкатеринбург — прекрасне місто, засноване у XVIII ст. Ваше завдання в цій задачі — дати нам інформацію про точний рік її заснування. Щоб завдання було менш обтяжливим, вам пропонується лише обчислити одну з чотирьох цифр року.

Для позначення позиції цифри необхідно обчислити, цифри нумеруються від 1 до 4, від найбільш значущих до найменш значущих. Наприклад, для 2013 року цифра 1 — «2», цифра 2 — «0», цифра 3 — «1», а цифра 4 — «3».

Якщо вам не вдалося взяти з собою енциклопедію, інформацію про Єкатеринбург можна отримати шляхом уточнення.

Вхідні дані Один рядок, який містить ціле число D ($1 \leq D \leq 4$), що вказує позицію цифри, яку ви повинні обчислити.

Вихідні дані Виведіть рядок з цілим числом, що представляє цифру в позиції D року заснування Єкатеринбурга.

Джерело
ACM ICPC Regional Latino America 2013, Warmup Session
Складність: 1% — 214/168/153/151.

Програма на **Python** (26 ms, 5.5 MiB)

```
print('1723'[int(input())-1])
```

3.10.2. e1607 Число у зворотньому порядку, e0947 Зворотній порядок, e0943 Перестановка цифр трицифрового

Записати ціле **невід'ємне** число **n** у зворотньому порядку.
Вхідні дані Одне ціле невід'ємне 64-х розрядне число.
Вихідні дані Запис числа у зворотньому порядку.
Складність: 15% — 13750/4596/4085/3469.

Програма на **Python** (20 ms, 5 MiB)

```
print(input()[::-1])
```

Розв'язок є і в [16].

Той же розв'язок в аналогічній задачі **e0947 Зворотній порядок**.

Записати дане трицифрове **натуральне** число в зворотному порядку.

Вхідні дані У єдиному рядку задане натуральне трицифрове число.
Вихідні дані Запис заданого числа у зворотному порядку.
Складність: 10% — 6267/3318/3272/2954.

Також цей же розв'язок в **e0943 Перестановка цифр трицифрового**.

У заданому трицифровому натуральному числі поміняти першу та останню цифри місцями.

Вихідні дані Одне натуральне трицифрове число n ($100 \le n \le 999$).
Вихідні дані Вивести число, отримане в результаті вказаного обміну.

Складність: 11% — 14994/7010/6392/5719.

Деякий розв'язок є в [16].

Задача **e1607 Число у зворотньому порядку** може бути розв'язана працюючи з числами. Читаємо число, виділяємо останню цифру і виводимо її. І так для всіх цифр числа.

Наведемо модифікований розв'язок з [16] на **C++** (3 ms, 1.8 MiB)

```
#include <iostream>
using namespace std;
int main() {
    long long n;
```

```
    cin >> n;
    do {
        cout << n%10;
        n /= 10;
    } while(n);
}
```

3.10.3. e1608 Число-паліндром

Перевірте, чи є задане число паліндромом.

Число називається паліндромом, якщо воно читається зліва направо і зправа наліво однаково.

Вхідні дані Одне невід'ємне ціле 32-х розрядне число.

Вихідні дані Вивести "**Yes**" якщо число паліндром, в іншому випадку "**No**".

Складність: 14% — 7099/2969/2639/2276.

Використовуємо, спираємось на 3.10.2.

Програма на **Python** (19 ms, 5,1 MiB)

```
n = input()
print('Yes') if n == n[::-1] else print('No')
```

3.10.4. a0173 Число-паліндром

Нагадаємо, що паліндромом зветься рядок, що однаково читається в обидві сторони. Наприклад, рядок «ABBA» є паліндромом, а рядок «ABC» — ні.

Необхідно визначити, в яких системах числення з основою від 2 до 36 подання заданого числа N є паліндромом.

В системах числення з основами більшими 10 в якості цифр використовуються літери англійської абетки: $A, B, ..., Z$. Наприклад, $A_{11} = 10_{10}$, $Z_{36} = 35_{10}$.

Вхідні дані

Вхідний файл INPUT.TXT містить задане число N в десятковій системі числення ($1 \le N \le 10^9$).

Вихідні дані

Якщо відповідні основи системи числення визначаються єдиним чином, то виведіть в першому рядку вихідного файлу OUTPUT.TXT слово «unique», якщо ж вони не єдині — виведіть в першому рядку вихідного

файлу слово «multiple». Якщо ж такої основи системи числення не існує — виведіть в в першому рядку вихідного файлу слово «none».

В випадку існування хоча б одної потрібної основи системи числення виведіть через пробіл у зростаючому порядку в другому рядку вихідного файлу всі основи систем числення, що задовільняють умовам.

Приклад

№	INPUT.TXT	OUTPUT.TXT
1	123	unique 6
2	111	multiple 6 10 36
3	102892748	none

Складність задачі: 29%, розв'язуваність 89% (2215).

Pascal (0.062 с, 1308 Кб)

```
var  n,i,j,k,l,t,p : longint;
     d,r : array [0..36] of byte;
begin
  read(n);
  k:=0;
  for i:=2 to 36 do
    begin
      l:=0; t:=n;
      repeat d[l+1]:=t mod i; t:=t div i; inc(l);
      until t=0;
      p:=1;
      for j:=1 to (l div 2) do
        if d[j]<>d[l-j+1] then p:=0;
      if p=1 then begin inc(k); r[i]:=1; end;
    end;
  case k of
     0 :  write  ('none');
     1 :  writeln('unique');
     else writeln('multiple');
  end;
  for j:=2 to 36 do if r[j]=1 then write(j,' ');
end.
```

3.10.5. e8243 Перша цифра числа

Знайти першу цифру цілого числа. Відлік починати з найвищого розряду.

Вхідні дані

Одне ціле 64-розрядне число, що містить не менше однієї цифри. Число може бути від'ємним.

Вихідні дані

Виведіть першу цифру заданого числа.

Автор Михайло Медведєв

Складність: 10% — 5028/2114/1853/1670.

Як і в попередній задачі простіше обробляти число як рядок, врахувавши що число може бути від'ємним (підказка в умові). Введемо ціле число, видалимо знак мінус, якщо він є ($abs()$), перетворимо в рядок ($str()$) і виведемо перший символ.

На **Python** (24 ms, 7.7 MiB)

```
print(str(abs(int(input()))))[0])
```

3.10.6. e1605 Друга цифра числа

Знайти другу цифру цілого числа. Відлік починати з найвищого розряду.

Вхідні дані

Одне ціле 64-розрядне число, що містить не менше двох цифр. Число може бути від'ємним.

Вихідні дані

Виведіть другу цифру заданого числа.

Складність: 13% — 13470/4764/4079/3546.

Задача аналогічна попередній, тільки виводимо другий символ (цифру) замість першої. Програма на **Python** (23 ms)

```
print(str(abs(int(input()))))[1])
```

Розв'язок є і в [16].

3.10.7. e1609 Кількість даних цифр в числі

Підрахувати кількість цифр a в числі n.

Вхідні дані

У першому рядку записано одне ціле 32-розрядне число n.
У другому рядку записано одну цифру a.
Вихідні дані Одне число — розв'язок задачі.

Input #1	**Output #1**
25557 5	3
Input #2	**Output #2**
100 0	2

Складність: 9% — 9809/4390/3759/3410.

Програма на **Python** (23 ms)

```
print(input().count(input()))
```

Деякі розв'язки є в [16].

3.10.8. e5628 Трицифрове число

Дано ціле трицифрове число.
Переставляючи цифри цього числа створіть найменш можливе трицифрове число.
Вхідні дані Одне ціле трицифрове число.
Вихідні дані Відповідь до задачі.

Input	**Output**
431	134

Складність: 41% — 2497/371/512/302.

Введене розіб'ємо на символи (перетворимо в список символів). Якщо число від'ємне (перший символ «-» мінус), видалимо перший символ, відсортуємо цифри в порядку спадання, додамо символ «мінус» і, ясна річ, виведемо. Якщо ж число додатнє, відсортуємо цифри в порядку зростання. Перед виведенням перевіримо спочатку цифру. Якщо це нуль, поміняємо з другою. Потім аналогічно поміняємо за потреби першу цифру з третьою.

Програма на **Python** (29 ms, 5.5 MiB)

```
n=list(input())
if n[0] == '-':
    n.pop(0)
    n.sort(reverse=1)
    n.insert(0,'-')
else:
```

```
    n.sort()
    if n[0] == '0': n[0],n[1]=n[1],n[0]
    if n[0] == '0': n[0],n[2]=n[2],n[0]
print(''.join(n))
```

3.10.9. e8896 Різні цифри

Програма повинна ввести з консолі ціле трицифрове число **N** та вивести у відповідь **YES**, якщо всі цифри числа **N** різні і **NO** у протилежному випадку.

Вхідні дані Ціле трицифрове число **N**.

Вихідні дані Відповідь до задачі.

Джерело Серія задач "Абетка програмування"

Складність: 12% — 1543/514/545/477.

Знаходимо множину цифр від'ємного/додатнього цілого числа. Якщо кількість елементів множини три, результат YES.

Програма на **Python** (18 ms, 5.1 MiB)

```
print('YES') if len(set(list(str(abs(int(input()))))))\
  ==3 else print('NO')
```

Ця програма в один рядок, розірваний для друку.

3.10.10. e7459 Непарні розряди

Знайти добуток цифр п'ятицифрового числа n, які стоять на непарних розрядах.

Вхідні дані Ціле п'ятицифрове число n.

Вихідні дані Вивести добуток цифр на непарних розрядах.

Складність: 28% — 5936/1690/1719/1243.

Python (29 ms, 7.7 MiB)

```
n=str(abs(int(input())))
print(int(n[0])*int(n[2])*int(n[4]))
```

Модифікований розв'язок з [16]

```
n=abs(int(input()))
print((n//10000)*(n//100%10)*(n%10))
```

3.10.11. e2396 Число на англійській

Задане натуральне число M вивести словами на англійській мові.

Вхідні дані Натуральне число M ($0 < M < 1000$).

Вихідні дані Записане прописом на англійській мові число M.

Примітка: Таблиця містить запис усіх необхідних англомовних числівників.

		10	ten				
1	one	11	eleven			100	one hundred
2	two	12	twelve	20	twenty	200	two hundred
3	three	13	thirteen	30	thirty	300	three hundred
4	four	14	fourteen	40	forty	400	four hundred
5	five	15	fifteen	50	fifty	500	five hundred
6	six	16	sixteen	60	sixty	600	six hundred
7	seven	17	seventeen	70	seventy	7000	seven hundred
8	eight	18	eighteen	80	eighty	800	eight hundred
9	nine	19	nineteen	90	ninety	900	nine hundred

Input	**Output**
725	seven hundred twenty five

Джерело II етап Всеукраїнської олімпіади в Житомирській області

Складність: 31% — 723/184/253/174.

Програма на **Python** (37 ms, 5.5 MiB)

```
u,s=['','one','two','three','four','five','six', \
   'seven','eight','nine'],[]
t=['ten','eleven','twelve','thirteen','fourteen',\
   'fifteen','sixteen','seventeen','eighteen', \
   'nineteen']
d=['twenty','thirty','forty','fifty','sixty', \
   'seventy','eighty','ninety']
m='%03d' % int(input())
if m[0] != '0':
    s.append(u[int(m[0])]);
    s.append('hundred')
if m[1] == '1': s.append(t[int(m[2])])
else:
    if m[1]!='0': s.append(d[int(m[1])-2])
    s.append(u[int(m[2])])
print('␣'.join(s))
```

3.10.12. e0963 Перестановка слів

Поміняйте в рядку ім'я і прізвище людини.

Вхідні дані Вхідний файл містить один рядок, у якому записані прізвище та ім'я людини (відокремлені рівно одним пропуском).

Вихідні дані У вихідний файл виведіть цю ж інформацію, проте спочатку ім'я, а потім прізвище, також відокремлені рівно одним пропуском.

Ліміт часу 0.1 с.

Складність: 8% — 2816/1599/1566/1442.

Python (22 ms, 5 MiB)

```
print(*(input().split()[::-1]))
```

Деякий розв'язок є і в [16].

3.10.13. e0959 Сума крайніх

Знайти суму крайніх цифр чотирицифрового натурального числа.

Вхідні дані Одне натуральне чотирицифрове число.

Вихідні дані Вивести суму крайніх цифр числа.

Складність: 3% — 5527/4247/3818/3718.

Python

```
n=input()
print(int(n[0])+int(n[len(n)-1]))
```

3.10.14. e0951 Обмін

Вхідні дані У єдиному рядку задане натуральне чотирицифрове число.

Вихідні дані Нове число.

Складність: 3% — 5059/3697/3387/3296.

Програма на **Python**

```
n=input()
print(n[0],n[2],n[1],n[3],sep='')
```

3.10.15. e9393 Видаліть непарні цифри

Дано рядок. Видаліть з нього всі непарні цифри.
Вхідні дані Один рядок, що містить букви і цифри.
Вихідні дані Виведіть результуючий рядок.
Складність: 13% — 217/122/125/109.

Програма на **Python** (20 ms; 5,1 MiB)

```
s,d,r=input(),['1','3','5','7','9'],''
for i in s:
    if not i in d: r += i
print(r)
```

3.10.16. e0909 Кількість слів

Визначити кількість слів у заданому фрагменті тексту.
Вхідні дані В єдиному рядку задано фрагмент тексту на англійській мові, кількість символів у якому не перевищує 250. Гарантується, що у тексті відсутні тире, дефіси, цифри і числа.
Вихідні дані Єдине число — кількість слів у фрагменті.
Складність: 10% — 12642/6228/5444/4888.

На **Python** (21 ms, 7,5 MiB)

```
print(len(input().split()))
```

На **PHP** (21 ms, 7,5 MiB)

```
<?php echo str_word_count(fgets(STDIN));
```

3.10.17. e0329 Кількість слів

Є деяке речення на невідомій мові. Порахувати кількість слів у ньому. Літерами алфавіту у невідомій мові є літери латинського алфавіту та арабські цифри. Гарантується, що інших символів, крім пропусків та розділових знаків у реченні нема.
Вхідні дані В одному рядку дано речення на невідомій мові.
Вихідні дані Вивести кількість слів у реченні.
Складність: 49% — 19396/2322/3621/1830.

На **Python**

```
w=input().split()
k=len(w)
for i in w:
    if i == '-': k -= 1
print(k)
```

На **PHP** (21 ms, 7,5 MiB)

```
<?php echo str_word_count(str_replace("-", "", \
  fgets(STDIN)));
```

3.10.18. e0912 Кількість речень

Визначити кількість речень у заданому фрагменті тексту.

Вхідні дані У єдиному рядку задано фрагмент тексту на англійській мові, кількість символів у якому не перевищує 250. Гарантується, що у тексті відсутні тире, дефіси, цифри і числа.

Вихідні дані Єдине число — кількість речень у фрагменті.

Складність: 29% — 14160/3270/3823/2732.

Речення закінчуються на **.!?**

На **Pascal** (24 ms, 5.1 MiB)

```
var s:string;
    i,k:byte;
begin
 read(s);
 for i:=2 to length(s)-1 do
  if (s[i] in ['.','!','?']) and (s[i+1]=' ') then inc(k);
 if s[length(s)] in ['.','!','?'] then inc(k);
 writeln(k)
end.
```

3.10.19. e0494 Голосні

До голосних літер в латинському алфавіті відносяться літери **A, E, I, O, U** і **Y**. Інші літери вважаються приголосними. Напишіть програму, яка підраховує кількість голосних літер в тексті.

Вхідні дані У вхідному файлі міститься один рядок тексту, який складається лише із заглавних латинських літер та проміжків. Довжина рядка не перевищує 100 символів.

Вихідні дані У вихідний файл вивести одне ціле число — кількість голосних у вхідному тексті.

Складність: 6% — 9612/4752/4273/3999.

На **Python** [16]

```
s,n=input(),0
for i in 'AEIOUY': n += s.count(i)
print(n)
```

3.10.20. e4722 Квадрат числа

Число n записали k разів підряд. Отримане число піднесли до квадрату.

Скільки вийшло?

Вхідні дані У першому рядку записано ціле невід'ємне число n ($n \leq 777$). У другому рядку записано ціле додатнє число k ($k \leq 777$).

Вихідні дані Виведіть число, яке отримали у результаті описаних вище дій.

Складність: 25% — 1477/576/700/523.

Python (43 ms, 5.5 MiB)

```
print(int(input()*int(input()))**2)
```

Розв'язок є і в [16].

3.10.21. e5049 Видали пропуски

Задано рядок. Вам потрібно перетворити усі пропуски, що йдуть підряд, в один.

Вхідні дані Один рядок символів з довжиною не більше 1000.

Вихідні дані Виведіть змінений рядок.

Складність: 15% — 1655/715/660/559.

Python (41 ms, 5.4 MiB)

```
s=input()
while s.find('␣␣')>-1: s=s.replace('␣␣','␣')
print(s)
```

Розв'язок є і в [16], однак не враховуються пробіли перед текстом.

Модифікуємо наведений там найкращий розв'язок

```
print('␣'.join(input().split()))
```

На відміну від попередньої програми тут видаляються початкові та кінцеві пробіли.

3.10.22. e8610 Попередня і наступна буква

Дана буква англійського алфавіту. Виведіть її попередню і наступну літери.

Вхідні дані Одна буква c ('A' < c < 'Z' або 'a' < c < 'z') англійського алфавіту (прописна або заголовна).

Вихідні дані Виведіть букву, наступну перед c і букву йде після c в англійському алфавіті (відповідно прописну або заголовну).

Складність: 9% — 1593/856/792/719.

Знаходимо ASCII код символу та символ за його кодом.

У введенні можуть бути інші символи — пробіли. В мовах з посимвольним введенням останнє несуттєво.

Програма на **Python** (18 ms, 5.1 MiB)

```
c = ord(input().strip())
print(chr(c-1),chr(c+1))
```

3.10.23. e8571 Підрахувати букви

Задано рядок s і буква c. Скільки разів буква зустрічається в рядку?

Вхідні дані Перший рядок містить рядок **s** з не більше ніж 100 символами. Другий рядок містить маленьку букву латинського алфавіту **c**.

Вихідні дані Виведіть скільки разів буква **c** зустрічається в рядку **s**. Одна і та ж заголовна і прописна буква вважаються однаковими. Тобто '**a**' і '**A**' вважаються однаковими буквами.

Складність: 16% — 2773/1066/1022/861.

Переведемо введене в нижній регістр і порахуємо кількість потрібної літери. В другому рядку можуть бути пробіли після літери, при введені їх видалити (strip() або rstrip()).

Програма на **Python** (22 ms, 5.1 MiB)

```
print(input().lower().count(input().strip()))
```

3.10.24. e8570 Довжина слів

Заданий текст — послідовність слів. Знайдіть довжину кожного слова.

Вхідні дані Текст містить послідовність слів. Довжина кожного слова не більше 20.

Вихідні дані Для кожного слова в одному рядку виведіть його довжину.

Складність: 12% — 2208/1019/868/763.

Назва є оманливою, оскільки глибина рекурсії, як видно з умови задачі, може складати 999, що не допустимо при виконанні більшості коду. Можна побачити, що описана функція задає суму арифметичної прогресії (2.1.3) натуральних чисел.

Програма на **Python** (49 ms, 5.1 MiB)

```
t = []
for line in open('input.txt'):
    t.extend(map(len,input().split()))
print(*t)
```

Програма однакова з (3.6.3).

3.10.25. e9625 toUpperCase

Задана рядок символів. Перетворіть усі малі літери латинського алфавіту в заголовні.

Вхідні дані Рядок, що складається з не більше ніж 100 символів.

Вихідні дані Виведіть рядок символів, в якій усі малі літери латинського алфавіту перетворені в заголовні.

Автор Михайло Медведєв

Складність: 4% — 137/92/92/88.

На **Python** (25 ms, 5.1 MiB)

```
print(input().upper())
```

3.10.26. e0119 Степінь двійки

В рядку послідовно записані n степеней двійки, тобто числа від 2 до 2^n без проміжків. Знайдіть значення n.

Вхідні дані В одному рядку без проміжків записано n ($1 \le n \le 1000$) послідовних степеней двійки.

Вихідні дані Вивести значення n.

Автор Михайло Медведєв

Джерело III етап Всеукраїнської олімпіади з інформатики в Житомирській обл. 2006-2007 р

Складність: 25% — 6428/1869/2067/1540.

Враховуємо, що $2^{1000} \simeq 1.1 \times 10^{301}$, працюємо з довгою арифметикою.

На **Python**

```
s,n,k=input(),2,0
while len(s)>0: s=s[len(str(n)):]; n*=2; k+=1
print(k)
```

Схожий розв'язок є в [16].

3.10.27. e6598 Різні цифри

Мешканці Нлогонії дуже забобонні. Одне з їх переконань полягає в тому, що номери вуличних будинків, які мають повторну цифру, приносять невдачу мешканцям. Тому вони ніколи не житимуть у будинку, який на вулиці має номер, як 838 чи 1004.

Королева Нлогонії наказала побудувати новий приморський проспект і хоче присвоїти новим будинкам лише номери без повторних цифр, щоб уникнути дискомфорту серед її підданих. Її Величність призначила, щоб написати програму, яка з урахуванням двох цілих чисел N і M визначає максимальну кількість будинків, яким можна присвоїти номери вулиць між N і M, включно, які не мають повторних цифр.

Вхідні дані

Кожен тестовий випадок описаний за допомогою одного рядка. Рядок містить два цілих числа N і M, як описано ($1 \le N \le M \le 5000$).

Вихідні дані

Для кожного тестового випадку виведіть рядок з цілим числом, що представляє кількість номерів будинків вулиць між N і M включно, без повторних цифр.

Джерело ACM ICPC Regional Latino America 2012

Складність: 12% — 172/93/77/68.

Якщо кількість цифр номеру будинку збігається з кількістю елементів множини цих цифр, то — номер будинку має однакові цифри.

Програма на **Python** (895 ms, 5.5 MiB)

```
for line in open("input.txt"):
    n,m=map(int,line.split())
    k=0
    for j in range(n,m+1):
        if len(str(j))==len(set(str(j))): k+=1
    print(k)
```

3.10.28. e7234 Кондиціонер Степана

В офісі, де Степан працює програмістом, встановили кондиціонер нового типу. Цей кондиціонер відрізняється особливою простотою в управлінні. У кондиціонера є всього лише два керованих параметра: бажана температура і режим роботи.

Кондиціонер може працювати в наступних чотирьох режимах:
- «freeze» — охолодження. У цьому режимі кондиціонер може тільки зменшувати температуру. Якщо температура в кімнаті і так не більше бажаної, то він вимикається.
- «heat» — нагрів. У цьому режимі кондиціонер може тільки збільшувати температуру. Якщо температура в кімнаті і так не менше бажаної, то він вимикається.
- «auto» — автоматичний режим. У цьому режимі кондиціонер може як збільшувати, так і зменшувати температуру в кімнаті до бажаної.
- «fan» — вентиляція. У цьому режимі кондиціонер здійснює тільки вентиляцію повітря і не змінює температуру в кімнаті.

Кондиціонер досить потужний, тому при налаштуванні на правильний режим роботи він за годину доводить температуру в кімнаті до бажаної.

Потрібно написати програму, яка по заданій температурі в кімнаті t_{room}, встановленим на кондиціонері бажаної температурі t_{cond} і режиму роботи визначає температуру, яка встановиться в кімнаті через годину.

Вхідні дані Перший рядок вхідного файлу містить два цілих числа t_{room}, і t_{cond}, розділених рівно одним пропуском ($-50 \le t_{room} \le 50, -50 \le t_{cond} \le 50$). Другий рядок містить одне слово, записане малими літерами латинського алфавіту — режим роботи кондиціонера.

Вихідні дані Вихідний файл повинен містити одне ціле число — температуру, яка встановиться в кімнаті через годину.

Input	Output
10 20 heat	20

Пояснення до прикладів:

У першому прикладі кондиціонер знаходиться в режимі нагріву. Через годину він нагріє кімнату до бажаної температури в 20 градусів.

У другому прикладі кондиціонер знаходиться в режимі охолодження. Оскільки температура в кімнаті нижча, ніж бажана, кондиціонер самостійно вимикається і температура в кімнаті не поміняється.

Ліміт часу 0.1 секунд

Джерело III етап Всеукраїнської олімпіади 2014 р.

Складність: 10% — 1608/642/640/575.

Програма на **Python** (40 ms, 5.5 MiB)

```
r,c=map(int,input().split())
f=input()
if f=='freeze': print(min(r,c))
if f=='heat': print(max(r,c))
if f=='auto': print(c)
if f=='fan': print(r)
```

3.10.29. e6070 Рахуючи овець

Після довгої ночі кодування у Чарльза Пірсона Петерсона виникають проблеми зі сном. Це не тільки тому, що він все ще замислюється над проблемою, над якою працює, але також через те, що випив занадто багато кави протягом годин тижня. Це трапляється часто, тому Чарльз розробив процедуру рахування овець. Не тварин, а слів. Конкретно, він придумує список слів, багато з яких близькі за написанням до «sheep», а потім підраховує, скільки насправді є слів «sheep». Чарльз завжди обережний, щоб бути чутливим до регістру, тому "Sheep"не рахується. Ви повинні написати програму, яка допомагає Чарльзу рахувати «sheep».

Вхідні дані Введення складається з декількох тестових екземплярів. Перший рядок буде складатися з одного натурального цілого числа $n \leq 20$, що становить кількість тестових екземплярів. Вхід для кожного екземпляра буде в двох рядках. Перший рядок буде складатися з натурального цілого $m \leq 10$, а другий рядок буде складатися з m слів, розділених одним пробілом і кожен, що містить не більше 10 символів.

Вихідні дані Для кожного тестового екземпляра потрібно створити один рядок виводу у форматі:

Case i: This list contains n sheep.
Значення i — це номер екземпляра (вважаємо, що ми починаємо нумерацію з 1), а n — кількість разів, коли слово "sheep"з'являється у списку слів для цього екземпляра. Вихідні рядки повинні бути розділені одним порожнім рядком.

Джерело ACM ICPC East Central Regional Practice Contest 2000
Складність: 10% — 406/128/127/114.

Програма на **Python** (23 ms, 5.4 MiB)

```
for i in range(int(input())):
    input()
    print('Case␣'+str(i+1)+ \
      ':␣This␣list␣contains␣'+str(input().split().\
      count('sheep'))+'␣sheep.')
    if i<n-1: print()
```

3.10.30. e2164 Шифр Юлія

Юлій Цезар використовував свій спосіб шифрування тексту. Кожна літера мінялась на наступну за алфавітом через k позицій по колу. Необхідно за заданою шифровкою встановити початковий текст.

Вхідні дані У першому рядку задано шифровку, яка складається з не більш ніж 255 великих латинських літер. У другому рядку число k ($1 \le k \le 10$).

Вихідні дані Вивести результат розшифровки.

Складність: 8% — 5074/2616/2159/1978.

На **Pascal** (3 ms, 0.68 MiB)

```
var s:string;
    k,i:byte;
begin
   read(s,k);
   for i:=1 to length(s) do t:=
       write(chr((ord(s[i])-65-k+26)mod 26 +65));
end.
```

На **Python** (27 ms, 5.4 MiB)

```
s,k = input(),int(input())
```

```
for i in s: print(chr((ord(i)-65-k)%26+65),end='')
```

Схожий розв'язок є в [16].

3.10.31. e6767 Що сказала лисиця?

Нарешті вдалося виявити древню таємницю — звук лисиці. Ви пішли в ліс, озброївшись дуже хорошим цифровим диктофоном. Однак в лісі повно голосів тварин, і на Вашому записі може бути чути багато різних звуків. Але Ви добре підготовлені для виконання завдання: Ви точно знаєте всі звуки, що видають інші тварини. Тому решту запису — всі невстановлені шуми — повинні були зроблені лисицями.

Вхідні дані Перший рядок містить кількість тестів t. Перший рядок кожного тесту містить сам запис — слова англійського алфавіту, розділені пропуском. Кожен рядок містить не більше 100 літер і не більше 100 слів. Наступні кілька рядків містять інформацію про інших тварин в форматі **тварина goes звук**. Всього не більше 100 тварин, їх назви містять не більше 100 літер і задаються по-англійські. Рядка **fox goes ...** серед них немає.

Останній рядок тесту є питанням, на який Вам слід відповісти: що сказала лисиця?

Вихідні дані Для кожного тесту виведіть у окремому рядку звуки, зроблені лисицею, в порядку їх запису. Вважайте, що лисиця не мовчить (всупереч поширеній думці, лисиці не спілкуються азбукою Морзе).

Input	**Output**
1 toot woof wa ow ow ow pa blub blub pa toot pa blub pa pa ow pow toot dog goes woof fish goes blub elephant goes toot seal goes ow what does the fox say?	wa pa pa pa pa pa pow

Джерело 2013 ACM Central Europe, Ноябрь 15-17, Задача B
Складність: 29% — 715/182/213/152.

Видаляємо по черзі відомі звуки.

Програма на **Python** (32 ms, 5.5 MiB)

```
for i in range(int(input())):
    a = input().split()
    while 1:
        g = input()
```

```
        if g == 'what does the fox say?': break
        w = g.split()
        while w[2] in a: a.remove(w[2])
    print(' '.join(a))
```

3.10.32. e6827 Aaah!

Джон Маріус так сильно кричав на недавньому концерті Джастіна Байбера, що тепер йому потрібен докторе із-за болю в горлі. Доктор прописав йому говорити «aaah». На жаль, говорити Джону «aah» не завжди получалось. Деякі доктори пропонували говорити «aah», деякі вимагали «aaaaaah», а деякі просто хотіли досліджували горло Джона, проголошенням «h». (Діагноз часто був невірний, але питання не про це). Маріус не хотів ходити до докторів, так як вважав це порожньою втратою часу, тому хотів сам порівнювати свої можливості вимовляти «aaah» з вимогою доктора. Хіба хто-небудь хоче вимовляти «aaah», якщо доктор вимагає "aaaaaah"?

Кожен день Джон Маріус дзвонив різним докторам і питав, як довго належить говорити «aaah». Визначіть, чи витратив час дарма Джон у чергового доктора.

Вхідні дані

Складається з двох рядків. Перший рядок містить фразу «aaah», яку Джон Маріус здатний сказати сьогодні. Другий рядок містить фразу «aah», яку хоче почути доктор. Кожен рядок містить лише великі літери 'a' і 'h'. Кожне слово містить від 0 до 999 включно букв 'a', після чого стоїть одна буква 'h'.

Вихідні дані Вивести «go», якщо Маріус може піти до лікаря, і «no» в іншому випадку.

Input 1	**Output 1**
aaah aaaaah	no
Input 2	**Output 2**
aaah ah	go

Джерело 2012 Nordic Collegiate Programming Contest, Жовтень 6, Задача A

Складність: 6% — 4000/2037/1736/1628.

Порівнюємо рядки введення.

Python (123 ms, 7.9 MiB)

```
print('no') if input()>input() else print('go')
```

3.10.33. e4737 Видалення зайвих пропусків

Задано рядок. Напишіть програму, яка видалить з цього рядка усі зайві пропуски. Пропуск будемо вважати зайвим, якщо:

. він знаходиться на самому початку рядка, до самого першого слова;

. він знаходиться у кінці рядка, після самого останнього слова;

. декілька пропусків розміщено між словами (простіше кажучи, якщо слова відокремлено більше ніж одним пропуском, тоді усі пропуски, крім одного, зайві).

Вхідні дані Задано рядок s ($0 \leq |\ s\ | \leq 255$). Рядок містить лише латинські літери та пропуски.

Вихідні дані Виведіть рядок без зайвих пропусків.

Складність: 10% — 1176/685/653/588.

Найелегантніший розв'язок — використання вбудованого розщеплення (split()) в Python, при цьому зайві пробіли і видаляються.

На **Python** (23 ms, 5.1 MiB)

```
print(*input().split())
```

3.10.34. e7326 Спальні вагони

Поїзд складається із спальних вагонів, які позначаються буквою **k**, і сидячих вагонів, які позначаються буквою **p**. Знайдіть найбільшу кількість спальних вагонів, які розміщені один за одним в поїзді.

Вхідні дані В одному рядку записано послідовність букв **k** і **p**, довжина якої від 1 до 1000 символів.

Вихідні дані Вивести одне число — найбільшу кількість спальних вагонів, що слідують один за одним.

Input	**Output**
kpkkp	2

Джерело

2014 ACM-ICPC Україна, 2-й Раунд, Вересень 13, Задача C

Складність: 13% — 5773/2224/2016/1759.

На **Python** (24 ms, 5.4 MiB)

```
s=input()
k=len(s)
while s.find('k'*k)==-1: k -= 1
print(k)
```

3.10.35. e6052 Швидка сума

Швидка сума враховується по пакету даних, і є просто числом. Якщо дані поміняються, то швидка сума також змінюється. Тому швидка сума використовується для знаходження помилок при передачі даних, підтвердження вмісту документів, а також в багатьох інших ситуаціях, де необхідно виявити небажані зміни даних.

Вам належить реалізувати алгоритм обчислення швидкої суми, що зветься Quicksum. Пакет Quicksum опрацьовує тільки заголовні літери та пробіли. Текст завжди починається і завершується заголовною буквою. В інших місцях пробіли і букви можуть застрічатись в будь-яких комбінаціях, включно послідовні пробіли.

Quicksum дорівнює сумі добутку позицій символів на їх символьне значення. Значення пробілів дорівнює нулю, значення букв дорівнює їх позиціям в алфавіті. Так A = 1, B = 2 и так далі до Z = 26. Розглянемо приклади Quicksum для пакетів «ACM» і «**MID CENTRAL**»:

ACM: $1*1+2*3+3*13=46$

MID CENTRAL: $1*13+2*9+3*4+4*0+5*3+6*5+7*14+8*20+9*18+10*1+11*12=650$

Вхідні дані Складаються з одного або декількох пакетів. Останній рядок вхідних даних містить # і не обробляється. Кожен пакет знаходиться в окремому рядку, не починається і не завершується пробілом, і містить від 1 до 255 символів.

Вихідні дані

Для кожного пакету вивести в окремому рядку шукану швидку суму.

Джерело 2006 ACM North America, Mid-Central, Problem A

Складність: 5% — 386/229/202/191.

На **Python** (27 ms, 5.4 MiB)

```
while 1:
    s,q=input(),0
    if s=='#': break
    for i in range(len(s)):
        if s[i]!='␣': q+=(i+1)*(ord(s[i])-64)
```

```
print(q)
```

3.10.36. e7340 Поле-чудес

Петрик і Марічка захопились грою поле-чудес: Марічка записує слово, що складається з великих англійських букв, а Петрик старається розпізнати його, причому відгадана буква відкривається на всіх позиціях, де вона міститься. За яку найменшу кількість ходів Петрик зможе відгадати задане слово.

Вхідні дані Слово записане великими англійськими буквами (не більше 100 символів).

Вихідні дані Відповідь до задачі.

Джерело II етап Всеукраїнської олімпіади в Житомирської області

Складність: 12% — 3487/1842/1709/1504.

Фактично потрібно знайти множину літер слова та її кількість елементів.

Програма на **Pascal**

```
var w:string;
    i,k:byte;
    c:array['A'..'Z'] of byte;
    j:char;
begin
  read(w);
  for i:=1 to length(w) do c[w[i]]:=1;
  for j:='A' to 'Z' do k:=k+c[j];
  writeln(k)
end.
```

Програма на **Python**

```
print(len(set(list(input()))))
```

Розв'язок є і в [16], на C++ в [25].

3.10.37. e0622 Одиниці

На уроках інформатики вас, напевно, вчили переводити числа з одних систем числення у інші і виконувати інші подібні операції. Прийшов час

продемонструвати ці знання. Знайдіть кількість одиниць у двійковому запису заданого числа.

Вхідні дані У вхідному файлі міститься єдине ціле число n ($0 \leq n \leq 2\,000\,000\,000$).

Вихідні дані Вихідний файл повинен містити одне число — кількість двійкових одиниць у запису числа n.

Складність: 10% — 6473/3395/2831/2562.

Програма на **C++**

```
#include <iostream>
using namespace std;
int main() {
    int k, n;
    cin >> n;
    while (n > 0){k += n%2; n /= 2;}
    cout << k;
}
```

Програма на **Python** (22 ms, 5.5 MiB)

```
print(bin(int(input())).count('1'))
```

Розв'язки є і в [16], [25].

3.10.38. e1427 Калькулятор

У студента Васі є молодший брат Петя, який пішов до першого класу і почав вивчати арифметику. Додому у першому класі задали розв'язати багато прикладів на додавання та віднімання. Петя попросив Васю перевірити домашнє завдання. Побачивши дві сторінки написаних каракулями прикладів, Вася прийшов в жах від об'єму роботи і вирішив навчити Петю використовувати для самоперевірки комп'ютер. Для цього Васі потрібно написати програму, яка обчислювала б розв'язки потрібних арифметичних прикладів.

Вхідні дані Один рядок, у якому можуть зустрічатись цифри та символи «+» і «-». Довжина рядка не перевищує 10000 символів, значення всіх чисел у ньому не перевищують 10000.

Вихідні дані Вивести одне ціле число — результат обчислень.

Складність: 8% — 3448/1984/1637/1498.

Текст введення можна використати як команди (програму).

Python найкращий розв'язок з [16]

```
print(eval(input()))
```

Інші задачі на рядки

e1966 Великий плюс Умову та код наведено в 3.3.3.
e1119 Піраміда з символів Умова та код наведені в 3.3.4.
e1340 Алмаз Умова та код наведені в 3.3.6.
e1124 Алфавітне графіті Умова та код наведені в 3.3.21.
e1603 Сума цифр числа Умова та розв'язок наведені в 3.4.13.
e0007 Римські числа Умова та код наведені в 3.4.56.
e0903 Перша чи остання? Умова та розв'язок наведено в 3.4.8.

3.11. Порівняння, розгалуження

3.11.1. e8608 sgn функція

Знайдіть значення *sgn* функції:

$$sgn(x) = \begin{cases} 1, & x>0 \\ 0, & x=0 \\ -1, & x<0 \end{cases}$$

Вхідні дані Одне ціле число x ($-10^9 \le x \le 10^9$).
Вихідні дані
Виведіть значення *sgn* функції для заданого значення x.
Складність: 2% — 1206/717/648/633.

Програма на **Python** (20 ms, 5.1 MiB)

```
x = int(input())
if x>0: sgn = 1
elif x<0: sgn = -1
else: sgn = 0
print(sgn)
```

3.11.2. a0025 Більше-менше

Одна з основних операцій з числами — їх порівняння. Маємо підозру, що Ви в досконалості володієте цією операцією і можете порівнювати довільні числа, в тому числі і цілі. В цій задачі необхідно порівняти два цілих числа.

Вхідні дані

В двох рядках вхідного файлу INPUT.TXT записані числа A і B, що не перевищують за абсолютною величиною 10^9.

Вихідні дані

Запишіть у вихідний файл OUTPUT.TXT один символ «<», якщо A < B, «»>, якщо A > B і «=», якщо A=B.

Складність задачі: 3%, розв'язуваність 95% (66641)

Пам'ятаємо особливості введення і пишемо простий код.

Pascal (0,03 с, 580 Кб)

```
var a,b:int64;
begin
  read(a,b);
  if a>b then write('>');
  if a<b then write('<');
  if a=b then write('=');
end.
```

Python (0,046 с, 678 Кб)

```
a, b = int(input()), int(input())
if a > b: print('>')
if a < b: print('<')
if a == b:print('=')
```

3.11.3. e8873 Одноцифрове число

Задано ціле число **n**. Вивести **Ok**, якщо число **n** одноцифрове та **No** у протилежному випадку.

Вхідні дані Одне ціле число **n**.

Вихідні дані

Вивести **Ok**, якщо число **n** одноцифрове, та **No** інакше.

Автор Матвійчук Сергій Володимирович

Джерело Серія задач "Абетка програмування"

Складність: 13% — 3844/1293/1290/1127.

Використовуємо, посилаємось на 3.5.3.

На **Python** (19 ms, 5.1 MiB)

```
print('Ok') if len(str(abs(int(input())))) == 1 \
  else print('No')
```

3.11.4. e8242 Додатнє, від'ємне чи нуль

Задано ціле число n. Виведіть, чи є воно додатним, від'ємним або дорівнює 0.

Вхідні дані Одне ціле число n, за модулем не більше за 10^9.

Вихідні дані Виведіть «Positive», «Negative» чи «Zero» в залежності від значення n.

Автор Михайло Медведєв

Складність: 4% — 6790/3793/3076/2947.

На **C++** (2 ms, 1.76 MiB)

```
#include <iostream>
using namespace std;
int main(){
    int n;
    cin >> n;
    if (n > 0) cout << "Positive";
    else if (n < 0) cout << "Negative";
    else cout << "Zero";
}
```

3.11.5. e6012 Time Limit Exceeded

Як Ви вже повинні знати, один із вердиктів, які ви можете отримати, подаючи програму задачі, — це перевищення часового обмеження (TLE). Це означає, що час роботи Вашого розв'язку перевищує обмеження, встановлене суддями.

Припустимо, що тестуючий сервер може робити 100 000 000 операцій в секунду. З огляду на часову складність Вашого розв'язку, виражену за допомогою O-нотації, максимального розміру введення на тестовий випадок N, кількості тестових випадків T та часового обмеження для всіх випадків L, чи може Ваш розв'язок вкластись в час?

Припустимо, що Ваш розв'язок використовує лише прості операції та ігноруйте будь-які інші накладні витрати (наприклад, введення/виведення).

Вхідні дані

Введення починається з рядка, що містить ціле число C ($1 \le C \le 100$), кількість тестових випадків. Далі C рядків, кожен в форматі

N T L

де N, T і L — цілі числа, як описано в умові задачі ($1 \le N \le 1\,000\,000$, $1 \le T, L \le 10$)? що ідуть за одним із наступних:

$O(N)$, $O(N\hat{}2)$, $O(N\hat{}3)$, $O(2\hat{}N)$, $O(N!)$

Примітка. Ми використовуємо тут дуже спрощену модель складності, і ознайомлення з нотацією big-O не потрібно (або навіть може бути згубним). Просто припустимо, що застосування N до функції в круглих дужках — це те, що дає вам загальну кількість операцій, які використовуватиме ваше рішення.

Вихідні дані

Для кожного тестового випадку виведіть в окремому рядку або «TLE!» якщо час роботи рішення перевищує обмеження часу для цього тестового випадку або «May Pass», якщо ні.

Input	**Output**
5	May Pass.
O(N) 1000 10 10	TLE!
$O(2\hat{}N)$ 1000 10 10	May Pass.
O(N!) 2 10 10	May Pass.
$O(N\hat{}3)$ 1000 1 10	TLE!
$O(N\hat{}3)$ 1001 1 10	

Автор Hichem Zakaria Aichour

Джерело 2013 Calgary Collegiate Programming Contest, Problem A

Складність: 0% — 16/8/8/8.

На **Python** (31 ms, 5.2 MiB)

```
from math import factorial
m,tl = 'May␣Pass.','TLE!'
for i in range(int(input())):
  s=input().split()
  n,t,l=map(int,s[1:])
  l *= 100000000
  if s[0]=='O(N)':
    print(tl) if n*t>l else print(m)
  elif s[0]=='O(N^2)':
    print(tl) if n>100000 or n**2*t>l else print(m)
```

```
  elif s[0]=='O(N^3)':
    print(tl) if n>1000 or n**3*t>l else print(m)
  elif s[0]=='O(2^N)':
    print(tl) if n>32 or 2**n*t>l else print(m)
  else:
    print(tl) if n>12 or factorial(n)*t>l else print(m)
```

3.11.6. a0163, e7411 Рівняння для 5 класу!

Рівняння для п'ятикласників представляє собою рядок довжиною 5 символів. Другий символ рядка є або знаком '+' (плюс) або '-' (мінус), четвертий символ — знак '=' (дорівнює). З першого, третього і п'ятого символів рівно два є цифрами з діапазону від 0 до 9, і один - буквою **x**, яка позначає невідоме.

Потрібно написати програму, яка дозволяє вирішити дане рівняння відносно **x**.

Вхідні дані Один рядок, в якому записано рівняння.

Вихідні дані Виведіть ціле число — значення **x**.

Джерело ACM-ICPC Ukraine 2015, I етап Україна, 25 квітня 2015 року

Складність (a0163): 25%, розв'язуваність 90% (8221).

Складність (e7411): 14% — 2287/839/872/748.

Програма на **Python** (e7411 2 ms, 0.65 MiB)

```
s=input()
m=1 if s[1]=='+' else -1
if s[0]=='x': x=int(s[4])-m*int(s[2])
if s[2]=='x': x=m*(int(s[4])-int(s[0]))
if s[4]=='x': x=int(s[0])+m*int(s[2])
print(x)
```

Програма на **Pascal** (a0163 0.071 c, 56 Кб)

```
var   a,b : shortint;
      er : integer;
      s : string;
function sgn(q:char):shortint;
begin
  if q='+' then sgn:=1 else sgn:=-1;
end;
```

```
begin
  read(s);
  if s[1]='x' then
    begin
    val(s[3],a,er); val(s[5],b,er);
    write(b-sgn(s[2])*a); exit;
    end;
  if s[3]='x' then
    begin
    val(s[1],a,er); val(s[5],b,er);
    write((b-a)*sgn(s[2])); exit;
  end;
  if s[5]='x' then
    begin
    val(s[1],a,er); val(s[3],b,er);
    write(a+sgn(s[2])*b); exit;
  end;
end.
```

3.12. Бітові операції

3.12.1. e2802 Бітове подання

Вам задано число у десятковій системі числення, а ви повинні вивести його у двійковій системі числення.

Вхідні дані У кожному рядку задано невід'ємне ціле десяткове число n ($n < 1000$). Вхідні дані продовжуються до кінця файлу.

Вихідні дані Для кожного числа, отриманого на вході, вивести його подання у двійковій системі числення (без ведучих нулів).

Автор Анатолій Присяжнюк

Джерело

II етап Всеукраїнської олімпіади школярів 2012-2013, м. Бердичів

Складність: 20% — 2592/714/728/585.

Програма на **Python** (50 ms, 5,5 MiB)

```
while 1:
    try: print(bin(int(input()))[2:])
    except: break
```

```
for i in open('input.txt'): print(bin(int(input()))[2:])
```

3.12.2. e5050 Степінь двійки

Виведіть число 2^n, використовуючи лише бітові операції.

Вхідні дані Одне число n ($0 \le n \le 30$).

Вихідні дані

Виведіть число 2^n, використовуючи лише бітові операції.

Складність: 3% — 2707/2017/1703/1653.

Програма на **Python**

```
print(1<<int(input()))
```

3.12.3. e5314 2^k + 2^n, e2733 Сума степенів двійки

Задано два різних числа k та n. Виведіть значення $2^k + 2^n$, використовуючи лише бітові операції.

Вхідні дані Два різних числа k та n ($0 \le k, n \le 30$).

Вихідні дані Виведіть число $2^k + 2^n$.

Джерело 2010 ЛКШ Берендеєві Поляни, Серпень, Паралель C, День 1, Задача A

Складність: 5% — 3785/2419/2066/1973.

Програма на **Python**

```
n,m=map(int,input().split())
print((1<<n)+(1<<m))
```

Аналогічний розв'язок на C++ є в [16].

e2733 Сума степенів двійки Задано два цілих невід'ємних числа $n, m < 30$. n і m різні.

Виведіть значення суми $2^n + 2^m$, використовуючи лише бітові операції.

Вхідні дані Задані 2 числа.

Вихідні дані Шуканий результат.

Складність: 4% — 2820/1852/1578/1510.

3.12.4. e1612 Змініть одиницю

Задано число n. Використовуючи лише бітові операції, замінити першу праворуч одиницю у двійковому запису на нуль.

Вхідні дані Одне число n ($1 \le n \le 10^8$).

Вихідні дані Виведіть одне змінене число.

Складність: 7% — 2618/1617/1409/1312.

Зсунемо двійкові розряди вправо, вліво доку число не поміняється.

Програма на **Python**

```
n,k=int(input()),1
while n==n>>k<<k: k+=1
print(n>>k<<k)
```

3.12.5. e5315 Встановити біт

Задано цілі числа a та k. Виведіть число, яке отримується з a встановленням значення k-го біта в 1.

Вхідні дані В одному рядку задано два числа a та k ($0 \le a \le 10^9$).

Вихідні дані Виведіть число a зі встановленим k-им бітом.

Джерело 2010 ЛКШ Берендеєві Поляни, Серпень, Паралель C, День 1, Задача B

Складність: 6% — 2768/1489/1333/1247.

Програма на **Python**

```
a,k=map(int,input().split())
print(a|(1<<k))
```

Аналогічний розв'язок на C++ є в [16].

3.12.6. e5316 Інвертувати біт

Задано цілі числа та k. Виведіть число, яке отримується з інвертуванням k-го біта.

Вхідні дані В одному рядку задано два числа a та k ($0 \le a \le 10^9$).

Вихідні дані Виведіть число a з інвертованим k-им бітом.

Джерело 2010 ЛКШ Берендеєві Поляни, Серпень, Паралель C, День 1, Задача C

Складність: 8% — 2625/1474/1338/1230.

Програма на **Python**

```
a,k=map(int,input().split())
print(a^(1<<k))
```

Аналогічний розв'язок на C++ є в [16].

3.12.7. e5317 Значення біта

Дано цілі числа a та k. Виведіть значення k-го біта числа a, що дорівнює 0 або 1.

Вхідні дані В одному рядку задано два числа a та k ($0 \le a \le 10^9$).

Вихідні дані Виведіть значення k-ого біта числа a.

Джерело 2010 ЛКШ Берендеєві Поляни, Серпень, Паралель C, День 1, Задача D

Складність: 10% — 3314/1576/1379/1242.

Програма на **Python**

```
a,k=map(int,input().split())
print((a >> k) & 1)
```

Аналогічний розв'язок на C++ є в [16].

3.12.8. e5318 Обрізати старші біти

Задано ціле число a та натуральне число k. Виведіть число, яке складається лише з k останніх біт числа a (тобто обнуліть усі біти числа a, крім останніх k).

Вхідні дані

В одному рядку знаходиться два числа a та k ($0 \le a \le 10^9$).

Вихідні дані Виведіть число a з обнуленими бітами, крім останніх k.

Джерело 2010 ЛКШ Берендеєві Поляни, Серпень, Паралель C, День 1, Задача E

Складність: 6% — 2379/1409/1235/1156.

Програма на **Python**

```
a,k=map(int,input().split())
print(a^(a>>k<<k))
```

```
a,k=map(int,input().split())
print(a&int('1'*k,2))
```

Розв'язок на C++ є і в [16].

3.12.9. e5319 Обнулити біт

Задано цілі числа a та k. Виведіть число, яке отримується з a скиданням значення k-го біта в 0. Молодший біт має номер 0.

Вхідні дані Два числа a та k ($0 \le a \le 10^9$).

Вихідні дані Виведіть число з обнуленим k-им бітом.

Складність: 6% — 2709/1557/1315/1235.

Програма на **Python**

```
a,k=map(int,input().split())
print(a&~(1<<k))
```

Аналогічний розв'язок на C++ є в [16].

3.12.10. e1753 Молодший біт

Для заданого додатнього цілого A ($1 \le A \le 100$), вивести молодший біт A.

Наприклад, якщо $A = 26$, то його ми можемо записати у двійковому вигляді, як 11010, молодший біт A є 10, і на виході повинно бути 2.

Інший приклад виглядає наступним чином: при $A = 88$, це число A ми можемо записати у двійковій формі 1011000, молодший біт в A є 1000, і на виході повинно бути 8.

Вхідні дані Кожен рядок вхідних даних містить лише одне ціле число A ($1 \le A \le 100$). Рядок, який містить «0» позначає кінець уведення, і цей рядок не є частиною вхідних даних.

Вихідні дані Для кожного числа A, отриманого на вході, у окремому рядку вивести значення його молодшого біта.

Складність: 11% — 3142/1419/1270/1133.

Python (29 ms, 5.5 MiB)

```
while 1:
    a,b=int(input()),1
    if(not a): break
    while not a&b: b*=2
    print(b)
```

3.12.11. e5051 Швидке множення

Задано числа a та b. Ваша задача – вивести, використовуючи лише бітові операції та операції додавання і віднімання, число $x = (36 * a + b \ div \ 16) \ mod \ 32$.

Вхідні дані Вводяться два числа через пропуск. $0 \le a, b \le 10^6$.

Вихідні дані Вихідний файл повинен містити одне число $x = (36 * a + b \ div \ 16) \ mod \ 32$.

Складність: 1% — 530/446/411/405.

Python (29 ms, 5.4 MiB)

```
a,b=map(int,input().split())
print(((a&31)<<2)+(b>>4)&31)
```

3.12.12. e6311 Клавіатура

Молода інноваційна фірма розробила нову клавіатуру з ергономічним дизайном. У принципі, вона могла б бути сумісна з ОС Windows, оскільки у неї є клавіші Ctrl, Win, Alt, ScrLk, NumLock, CapsLock, Left Shift, Right Shift та ін. Ось лише інформацію про натисненість цих клавіш клавіатура передає у вигляді одного цілого числа у десятковій системі числення.

Напишіть програму, яка за заданим номером клавіші визначає, чи натиснена вона, якщо натиснена клавіша кодується одиничним значенням біта з відповідним номером у двійковому поданні числа, біти нумеруються справа наліво, починаючи з нульового.

Вхідні дані У першому рядку задано два цілих числа: n ($0 \le n \le 1024$) — код, отриманий від клавіатури, і, через не менше ніж 1 пропуск, m ($0 \le m < 10$) — номер клавіші, що перевіряється.

Вихідні дані Вивести **YES**, якщо клавішу натиснено, і **NO** у протилежному випадку.

Складність: 16% — 1179/591/553/465.

Визначаємо наявність потрібного біту. На **Python** (20 ms, 5.5 MiB)

```
n,m=map(int,input().split())
print('YES') if n&(2**m) else print('NO')
```

Також

e5868 A xor B Умова та код програми наведені в 3.1.7.

3.13. Обробка масивів

3.13.1. e0908 Ті, що діляться на 6

Для n цілих чисел визначити суму й кількість додатніх чисел, які діляться на 6 без остачі.

Вхідні дані У першому рядку задано кількість чисел n ($0 < n \leq 100$), у наступному рядку задано самі числа, значення яких по модулю не перевищують 10000.

Вихідні дані Виведіть кількість вказаних чисел та їх суму.

Джерело ДПА-2010 Варіант 8

Складність: 14% — 24165/7330/6989/6021.

На **Pascal**

```
var n,i,k:byte;
    a,s:int64;
begin
  read(n);
  for i:=1 to n do
    begin
      read(a);
      if ((a mod 6)>0)or(a<=0) then continue;
      s:=s+a;
      inc(k)
    end;
  writeln(k,' ',s)
end.
```

На **Python** (29 ms, 5.5 MiB)

```
input()
a = [int(s) for s in input().split()]
b=a.copy()
for i in b:
  if i<=0 or i%6>0: a.remove(i)
print(len(a),sum(a))
```

Як варіант

```
input(); s=k=0
a = list(map(int,input().split()))
for i in a:
```

```
    if i>0 and i%6==0: s += i; k += 1;
print(k,s)
```

3.13.2. e0928 Сума найбільшого та найменшого

Задано масив цілих чисел. Визначити суму найменшого та найбільшого елементів масиву.

Вхідні дані У першому рядку задано кількість елементів масиву n ($n \leq 100$). У другому рядку задано n елементів масиву, значення кожного з яких за модулем не перевищує 100.

Вихідні дані Вивести суму найменшого та найбільшого елементів масиву.

Джерело ДПА-2010 Варіант 28

Складність: 8% — 19105/9557/7140/6590.

На **Python** (22 ms, 5 MiB)

```
input()
a = [int(s) for s in input().split()]
print(max(a)+min(a))
```

3.13.3. e5713 Вітряна погода

П'ятачок лежачи на галявині спостерігав за травинками, які коливались. Він зрозумів, що коливаються вони тому, що дме вітер і відразу ж придумав, як за допомогою травинок можна виміряти силу вітру. *Силою вітру*, за визначенням П'ятачка, сьогодні вважається різниця між висотою найвищої та найнижчої травинок.

Вхідні дані У першому рядку знаходиться єдине число n — кількість травинок, за якими спостерігає П'ятачок. У наступному рядку задано через пропуск n чисел — висоти травинок.

Усі вхідні дані натуральні числа, які не перевищують 100, оскільки П'ятачок багато рахувати не любив і не вмів з дуже простої причини — він у своєму житті поки що не зустрічав чисел, більших за 100.

Вихідні дані Єдине число — сила вітру за визначенням П'ятачка.

Автор Анатолій Присяжнюк

Джерело 2-й етап Всеукраїнської олімпіади з інформатики 2013-2014 н.р. 8 кл. м. Бердичів

Складність: 6% — 4260/2395/2049/1925.

Python

```
input()
g = list(map(int,input().split()))
print(max(g)-min(g))
```

3.13.4. e0902 Рівень навчальних досягнень

Встановити рівень навчальних досягнень учня (початковий, середній, достатній, високий) відповідно до заданої оцінки (від 1 до 12).

Вхідні дані Одне число — бал учня.

Вихідні дані Вивести Initial для початкового рівня (оцінка від 1 до 3), Average для середнього (оцінка від 4 до 6), Sufficient для достатнього (оцінка від 7 до 9) і High для високого (оцінка від 10 до 12).

Джерело ДПА-2010 Варіант 2

Складність: 7% — 28943/14241/12014/11217.

На **Pascal**

```
const l:array[0..3] of string=('Initial','Average',
  'Sufficient','High');
var m:byte;
begin
  read(m);
  writeln(l[(m-1)div 3])
end.
```

На **Python** (39 ms, 5.1 MiB)

```
print(['Initial','Average','Sufficient','High']
  [(int(input())-1)//3])
```

3.13.5. e0923 Пора року

Визначити назву пори року за заданим номером місяця, використовуючи складені умови.

Вхідні дані Одне число — номер місяця.

Вихідні дані Для весняних місяців вивести **Spring**, для літніх — **Summer**, для осінніх — **Autumn** і для зимових — **Winter**.

Джерело ДПА-2010 Варіант 23

Складність: 7% — 27764/13089/11088/10322.

На **Pascal**

```
var n:byte;
begin
  read(n);
   case n of
    1,2,12: writeln('Winter');
    3..5  : writeln('Spring');
    6..8  : writeln('Summer');
    9..11 : writeln('Autumn');
  end
end.
```

На **Python** (39 ms, 5.1 MiB)

```
print(['Winter','Spring','Summer','Autumn'] \
  [int(input())%12//3])
```

3.13.6. e0907 Перший не більший за 2,5

Задано масив дійсних чисел. Визначити перший елемент масиву, який не перевищує 2.5.

Вхідні дані У першому рядку задано кількість елементів масиву n ($0 < n \leq 100$), у наступному рядку задано n дійсних чисел.

Вихідні дані Вивести в одному рядку спочатку індекс знайденого першого вказаного елемента масиву і через пропуск його значення з точністю 2 знаки після десяткової крапки. У випадку відсутності вказаного елементу в масиві вивести "Not Found"(без лапок).

Джерело ДПА-2010 Варіант 7

Складність: 14% — 23772/6703/6309/5404.

На **Pascal** (20 ms, 5.1 MiB)

```
var h,i:byte;
    a:real;
begin
  read(h);
  for i:=1 to h do
    begin
      read(a);
      if a>2.5 then continue;
      writeln(i,' ',a:0:2);
      exit
```

```
    end;
  writeln('Not Found')
end.
```

3.13.7. e1681 Суми цифр

Підрахувати кількість N-значних натуральних чисел, у яких суми цифр у двійковій і десятковій системах числення співпадають. (N = 1..10).

Вхідні дані В файлі записане натуральне число N (N = 1..10).

Вихідні дані Єдине число — відповідь до задачі.

Автор В.Л., Матвійчук С.В.

Джерело III етап Всеукраїнської олімпіади школярів 2010-2011, 1 тур, м. Житомир

Складність: 51% — 1143/199/340/165.

Аналогічно 3.4.58 для такої кількості вхідних даних окремо розраховуємо результати, заносимо в масив і в програмі виводимо потрібний елемент масиву.

На **Python** (53 ms, 5.1 MiB)

```
print([0,1,2,14,60,406,2256,13084,70978,423000,2556298]\
    [int(input())])
```

Розв'язок є і в [16].

Знайти потрібні числа схоже можна лише перебором всіх чисел, розрахунку сум їх цифр та підсумовуванні при співпадінні. Для $n = 10$ потрібно виконати дії для $\sim 10^{10}$ чисел. Програма на C++ витрачає на розв'язання більше ліміту часу 1 с вже для $n = 8$.

На **Python** розрахунок, як ми знаємо іде значно довше

```
n,k = int(input()),0
for i in range(10**(n-1),10**n):
    if sum(map(int,list(str(i))))==bin(i).count('1'):
        k += 1
print(k)
```

3.13.8. e8953 Вивести масив

Задано масив з n цілих чисел. Вивести його елементи в стовпчик, не змінюючи початковий порядок.

Вхідні дані Перший рядок містить число n ($1 \le n \le 100$). У другому рядку записані n цілих чисел, кожне з яких не перевищує за модулем 100.

Вихідні дані Вивести елементи масиву по одному числу в кожному рядку.

Автор Матвійчук Сергій Володимирович

Джерело Серія задач "Абетка програмування"

Складність: 4% — 1889/1051/978/934.

Програма на **Python** (19 ms, 5.1 MiB)

```
input()
print('\n'.join(list(input().split())))
```

3.13.9. e8954 Вивести масив 2

Задано масив з n цілих чисел. Вивести елементи масиву в одному рядку, змінивши початковий порядок на протилежний.

Вхідні дані Перший рядок містить число n ($1 \le n \le 100$). В наступних n рядках записані елементи масиву (по одному числу в кожному рядку), що не перевищують за модулем 100.

Вихідні дані Вивести елементи масиву в одному рядку в зворотному порядку.

Автор Матвійчук Сергій Володимирович

Джерело Серія задач "Абетка програмування"

Складність: 7% — 1262/531/520/485.

Читаємо, виводимо. Вхідні дані та система перевірки вимагають, щоб дані були цілими числами, тому прийшлось ввести перетворенні введеного в ціли числа.

Програма на **Python** (24 ms, 5.1 MiB)

```
a = []
for i in range(int(input())): a.append(int(input()))
print(*a[::-1])
```

3.13.10. e8955 Вивести масив 3

Задано масив з n цілих чисел. Виведіть тільки додатні його елементи, не змінюючи їх початковий порядок.

Вхідні дані Перший рядок містить число n ($1 \le n \le 100$). У другому рядку записані n цілих чисел, кожне з яких не перевищує за модулем 100.

Вихідні дані У першому рядку виведіть кількість додатних елементів масиву. У другому рядку виведіть самі додатні елементи. Якщо додатніх елементів в масиві немає, то виведіть "NO".

Автор Матвійчук Сергій Володимирович

Джерело Серія задач "Абетка програмування"

Складність: 14% — 2158/677/687/594.

Програма на **Python** (23 ms, 5.1 MiB)

```
n,t = input(),[]
a = list(map(int,input().split()))
for i in a:
    if i>0: t.append(i)
print(len(t),'\n',*t) if len(t) else print('NO')
```

3.13.11. e9617 Кількість додатних

Найдіть кількість додатних чисел в заданому масиві.

Вхідні дані Перший рядок містить кількість чисел n. Другий рядок містить n цілий чисел, за модулем не більше 100.

Вихідні дані Вивести кількість додатних чисел в масиві.

Автор Михайло Медведєв

Складність: 3% — 184/124/121/117.

В генераторі відмічаємо (1) додатні числа, підсумовуємо їх кількість.

Програма на **Python** (21 ms, 5.1 MiB)

```
input()
print(sum([1 if i>0 else 0 for i in \
      map(int,input().split())]))
```

3.13.12. e9618 Сума від'ємних

Найдіть суму від'ємних чисел в заданому масиві.

Вхідні дані Перший рядок містить кількість чисел n. Другий рядок містить n цілий чисел, за модулем не більше 100.

Вихідні дані Виведіть суму від'ємних чисел в масиві. Якщо від'ємних чисел в масиві немає, виведіть 0.

Автор Михайло Медведєв
Складність: 3% — 184/124/121/117.

Використовуємо, спираємось 3.13.11. В генераторі список в якому додатні числа замінюємо на 0. Підсумовуємо список і виводимо цей результат.

Програма на **Python** (29 ms, 5.1 MiB)

```
input()
print(sum([i if i<0 else 0 for i in \
      map(int,input().split())]))
```

3.13.13. e7365 Молоко та пиріжок

Учням першого класу призначають додаткову склянку молока та пиріжок, якщо першокласник важить менше 30 кг. В перших класах школи навчається n учнів. Склянка молока має об'єм 200 мл, а замовлені упаковки молока —- 0,9 л. Визначити кількість додаткових пакетів молока та пиріжків, необхідних щодня.

Вхідні дані У першому рядку задано ціле число n ($0 < n \leq 100$). У наступному рядку знаходяться n додатних дійсних чисел —- маси кожного першокласника.

Вихідні дані В одному рядку вивести два цілих числа — кількість додаткових пакетів молока та пиріжків, необхідних щодня.

Джерело II етап Всеукраїнської олімпіади з інформатики
Складність: 28% — 17504/3041/3363/2417.

В якості одиниці вимірювання візьмемо 100 мл. Тоді склянка молока (об'єм) — 2, пакет молока — 9. Працюємо з цілими числами. Додаємо 8, щоб заокруглити верх кількість пакетів (замість ceil()).

Програма на **Pascal**

```
var n,i,j,k:byte;
    m:real;
begin
  read(n);
  for i:=1 to n do
    begin
      read(m);
      if m<30 then inc(k);
    end;
```

```
  j:=(k*2)div 9;
  if ((k*2)mod 9)>0 then inc(j);
  writeln(j,' ',k)
end.
```

Програма на **C#**

```
using System;
namespace e7365{
  class Program    {
      static void Main(string[] args) {
          byte n = Convert.ToByte(Console.ReadLine());
          string[] w = Console.ReadLine().Split();
          int p=0;
          for(int i=0;i<n; ++i)
              if(Convert.ToDouble(w[i])<30) p++;
          Console.WriteLine("{0} {1}", (p*2+8)/9,p);
      }
  }
}
```

Програма на **Python**

```
input()
k = sum([1 if w<30 else 0 for w in \
    map(int,input().split())])
print((k*2+8)//9,k)
```

3.13.14. e7830 Найбільший елемент масиву

Дано масив з N цілих чисел. Знайти найбільший елемент масиву.

Вхідні дані В першому рядку записано число N. В наступному рядку записано N цілих чисел. Всі числа не перевищують 100.

Вихідні дані Відповідь до задачі

Складність: 9% — 3074/1520/1400/1281.

Python (32 ms, 7.7 MiB)

```
input()
print(max(map(int,input().split())))
```

Аналогічний розв'язок є і в [16], де є і розв'язок на C++.

3.13.15. e7831 Сума без максимального

Дано масив з n цілих чисел. Знайти суму всіх елементів масиву які не дорівнюють максимальному.

Вхідні дані В першому рядку записано число n ($n \leq 100$). В наступному рядку записано n цілих чисел, кожне з яких не перевищує за модулем 100.

Вихідні дані Відповідь до задачі

Складність: 9% — 7848/3326/2676/2443.

Найкращий розв'язок на **Python** з [16]

```
input()
a=list(map(int,input().split()))
m=max(a)
print(sum(a)-m*a.count(m))
```

Розв'язок є і в [16].

3.13.16. e7832 Кількість максимальних

Дано масив з n цілих чисел. Знайти кількість максимальних елементів масиву.

Вхідні дані В першому рядку записано число n ($n \leq 100$). В наступному рядку записано n цілих чисел, що не перевищують за модулем 100.

Вихідні дані Вивести кількість максимальних елементів масиву.

Складність: 6% — 4817/2763/2263/2127.

Python (32 ms, 7.7 MiB)

```
input()
a=list(map(int,input().split()))
print(a.count(max(a)))
```

Розв'язок є і в [16].

3.13.17. e7833 Більші за середнє арифметичне

Дано масив з n цілих чисел. Знайти суму та кількість чисел, які більші за середнє арифметичне елементів масиву.

Вхідні дані В першому рядку записано число n. В наступному рядку записано n цілих чисел. Усі числа не перевищують за модулем 100.

Вихідні дані Вивести суму та кількість чисел, які більші за середнє арифметичне елементів масиву.

Складність: 16% — 4550/1513/1529/1279.

На **Python** (22 ms, 7.7 MiB)

```
n=int(input())
a=list(map(int,input().split()))
m,s,k=sum(a)/n,0,0
for i in a:
    if i>m: s+=i; k+=1
print(s,k)
```

Схожий розв'язок є в [16].

3.13.18. e7834 Два найбільших

Дано масив з n цілих чисел. Знайти суму двох найбільших елементів масиву.

Вхідні дані В першому рядку записано число n ($n \leq 100$). В наступному рядку записано n цілих чисел, кожне з яких перевищує 100.

Вихідні дані Вивести суму двох найбільших елементів масиву.

Input 1	**Output 1**
5 1 5 2 6 3	11
Input 2	**Output 2**
4 1 4 4 1	8
Input 3	**Output 3**
10 11 10 10 10 10 10 10 10 10 10	21

Складність: 11% — 5886/2103/1863/1655.

З прикладів видно, що підсумовуються тільки два числа.

На **Python** (24 ms, 7.7 MiB)

```
n=int(input())
a=sorted(list(map(int,input().split())))
print(a[n-1]+a[n-2])
```

Схожий розв'язок є в [16].

3.13.19. e0113 Кульки

У продавця повітряних кульок є n кульок. Кожна з них має деякий колір. Але зовсім недавно Три Товстуни видали наказ, який дозволяє торгувати кульками тільки якогось одного кольору. Щоб не порушувати закон, але при цьому і не втратити прибуток, продавець вирішив перефарбувати деякі із своїх кульок.

Напишіть програму для визначення мінімальної кількості перефарбувань.

Вхідні дані В першому рядку задано кількість кульок n ($1 \le n \le 100\,000$). Другий рядок містить n цілих чисел, в межах від 1 до 9, що визначає колір кульок (1 — синій, 2 — зелений, 3 — голубий, 4 — червоний, 5 — рожевий, 6 — жовтий, 7 — сірий, 8 — чорний, 9 — білий).

Вихідні дані Виведіть мінімальну кількість кульок, які необхідно перефарбувати, щоб всі кульки були одного кольору.

Складність: 15% — 7480/2855/2757/2331.

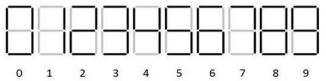
Не перефарбовувати потрібно тільки кульки, яких найбільше.

Програма на **Python** (70 ms, 6.4 MiB)

```
n,a=int(input()),[0]*10
b=map(int,input().split())
for i in b: a[i]+=1
print(n-max(a))
```

Схожий розв'язок є і в [16].

3.13.20. e0358 Прогрес в артилерії починається



Артилерія завжди була видом війська, у якому інтенсивно використовувались різноманітні обчислювальні пристрої. Колись вони були механічними і показували результати обчислень за допомогою коліщат з намальованими цифрами. Проте прогрес не стояв на місці. Одного разу конструктори розробили електричний пристрій, який показував результати за допомогою сегментних індикаторів — світлодіодів.

Для початку прогресу потрібно розробити програму, яка за заданим числом визначає кількість світлодіодів, які повинні загорітись, щоб вірно відобразити задане число.

Вхідні дані Задане число N ($0 \le N \le 10^9$).

Вихідні дані Шукана кількість світлодіодів.

Джерело
II етап Всеукраїнської олімпіади школярів 2008-2009, м. Бердичів
Складність: 9% — 2445/1128/1121/1018.

Створимо масив світлодіодів, що мають загорітись для кожної цифри (0—9). Підсумуємо кількість світлодіодів для всіх цифр заданого числа.

Програма на **Python** (21 ms, 5.1 MiB)

```
print(sum([[6,2,5,5,4,5,6,3,7,6][int(i)] for i in \
    input()]))
```

3.13.21. e2807 Кубики - 3

Дома у Вітека було 2 однакових набори кубиків з англійських літер, але під час чергового прибирання один з кубиків загубився. Допоможіть Вітеку визначити, який саме з кубиків відсутній у одному з наборів.

Вхідні дані У першому рядку задано кількість знайдених Вітеком кубиків n ($1 \le n \le 10^5$), а у другому рядку n символів, зображених на кожному з кубиків.

Вихідні дані Виведіть літеру, зображену на загубленому кубику, або повідомлення «Ok», якщо Вітек помилився і жоден з кубиків не загубився.

Автор Анатолій Присяжнюк
Джерело
II етап Всеукраїнської олімпіади школярів 2012-2013, м. Бердичів
Складність: 20% — 4159/1256/1242/995.

Python (27 ms, 7.7 MiB)

```
if int(input())%2==0: print('Ok'); exit()
s=input()
for j in s:
    if s.count(j)%2: print(j); break
```

3.13.22. e1356 SMS голосування

У фіналі фабрики зірок було проведено SMS голосування для визначення переможців серед N конкурсантів. Телеглядачі відправляли SMS з номером (число від 1 до N) свого улюбленого виконавця і кількість відповідних SMS склали рейтинг кожного учасника. Всього на головний комп'ютер конкурсу надійшло M повідомлень SMS. Потрібно скласти

програму, яка виведе номери трьох переможців у порядку спадання їх рейтингів та зростання номерів у випадку, якщо рейтинги рівні.

Вхідні дані У першому рядку записано два числа N і M ($3 \le N \le 100, 1 \le M \le 1\,000\,000$).

У наступному рядку M чисел, кожне з яких не перевищує N.

Вихідні дані Три числа — номери переможців записані в один рядок, через пропуск.

Складність: 41% — 1139/238/312/184.

Накопичуємо SMS в масиві на кожного учасника. Виводимо максимум, і так три рази. Помічаємо враховане число (-1 – число SMS — натуральне число, хоча, відповідно до умови може бути і 0).

На **Pascal** (97 ms, 0.68 MiB)

```
uses math;
var a:array[1..100] of int32;
    n,m,i,j,v:int32;
begin
  readln(n,m);
  for i:=1 to m do begin read(v); inc(a[v]) end;
  for i:=1 to 3 do
    begin
      v:=maxvalue(a);
      for j:=1 to n do
        if a[j]=v then begin write(j,' '); a[j]:=-1;
          break end
    end
end.
```

На **Python**

```
n,m, = map(int,input().split())
s =list(map(int,input().split()))
v,r = [0]*(n+1),[0,0,0]
for i in s: v[i] += 1
for j in range(3): k=v.index(max(v)); r[j]=k; v[k]=0
print(*r)
```

Аналогічний розв'язок є в [16].

3.13.23. e0462 Клавіатура

Всім відомо, що з часом клавіатура зношується, і клавіші на ній починають залипати. Звичайно, деякий час таку клавіатуру ще можна використовувати, але для натиснень клавіш приходиться застосовувати більшу силу.

При виготовленні клавіатури відразу для кожної клавіші задається кількість натиснень, які вона повинна витримати. Якщо знати ці величини для клавіатури, що використовується, то для певної послідовності натиснених клавіш можна визначити, які клавіші в процесі їх використання зламаються, а які — ні.

Потрібно написати програму, яка визначає, які клавіші зламаються у процесі заданого варіанту експлуатації клавіатури.

Вхідні дані Перший рядок містить кількість клавіш n ($1 \le n \le 100$) на клавіатурі. Другий рядок містить n цілих чисел — c_1, c_2, ..., c_n, де c_i ($1 \le c_i \le 100000$) — кількість натиснень, які витримує i-та клавіша. Третій рядок містить ціле число k ($1 \le k \le 100000$) — загальна кількість натиснень клавіш, і останній рядок містить k цілих чисел p_j ($1 \le p_j \le n$) — послідовність натиснених клавіш.

Вихідні дані Вивести n рядків, які містять інформацію про справність клавіш. Якщо i-а клавіша зламалась, то i-ий рядок повинен містити слово «yes» (без лапок), якщо ж клавіша працездатна — слово «no».

Складність: 7% — 3726/3726/1711/1588.

Python (117 ms, 11.5 MiB)

```
input()
c=list(map(int,input().split()))
input()
p=list(map(int,input().split()))
for i in p: c[i-1] -= 1
for i in c: print('yes') if i<0 else print('no')
```

3.13.24. e7841 Непарні елементи

Дано послідовність з n цілих чисел. Виведіть всі його непарні елементи.

Вхідні дані В першому рядку записано число n. В наступному рядку записано n ($n \le 100$) цілих чисел, що за модулем не перевищують 100.

Вихідні дані Вивести усі непарні елементи послідовності у тому ж порядку як вони зустрічаються на вході.

Складність: 10% — 6629/2723/2300/2077.

Python (23 ms, 7.7 MiB)

```
input()
print(*[i for i in map(int,input().split()) if i%2])
```

Аналогічний розв'язок є в [16].

3.13.25. e7842 Парні індекси

Дано масив з **n** цілих чисел. Виведіть усі його елементи з парними індексами. Нумерація починається з 0.

Вхідні дані Перший рядок містить число **n**. Другий рядок містить **n** цілих чисел. Усі числа за модулем не перевищують 100.

Вихідні дані Виведіть всі елементи масиву з парними індексами.

Автор Матвійчук Сергій Володимирович

Джерело Серія задач "Абетка програмування"

Складність: 6% — 3189/1682/1562/1471.

Python (21 ms, 5.1 MiB)

```
input()
print(*input().split()[::2])
```

Аналогічний розв'язок є в [16].

3.13.26. e7843 Більші попереднього

Дано масив цілих чисел. Виведіть всі елементи, які більші попереднього елементу.

Вхідні дані В першому рядку записано кількість чисел n у масиві. В наступному рядку записано n цілих чисел. Усі числа за модулем не перевищують 100.

Вихідні дані Виведіть елементи масиву, які більші попереднього.

Складність: 8% — 3257/1577/1453/1342.

Python

```
n=int(input())
a=list(map(int,input().split()))
print(*[a[i] for i in range(1,n) if a[i]>a[i-1]])
```

Схожий розв'язок є в [16].

3.13.27. e7844 Сусіди одного знака

Дано масив з N цілих чисел. Вивести пари сусідніх елементів одного знаку. Якщо сусідніх елементів одного знака немає — не виводьте нічого.

Вхідні дані В першому рядку записано число N. В наступному рядку записано N цілих чисел. Всі числа за модулем не перевищують 100.

Вихідні дані Пари елементів одного знаку.

Input	**Output**
10	-8 -10
-8 -10 -2 1 -9 5 -9 0 0 -9	-10 -2

Складність: 11% — 3616/1418/1349/1194.

Добуток сусідів одного знаку — додатній.

Python (20 ms, 7.7 MiB)

```
n=int(input())
a=list(map(int,input().split()))
for i in range(n-1):
    if a[i]*a[i+1]>0: print(a[i],a[i+1])
```

Розв'язок є і в [16].

3.13.28. e7845 Більші своїх сусідів

Дано масив з n цілих чисел. Визначте, скільки в цьому масиві елементів, які більші двох своїх сусідів і виведіть кількість таких елементів. Крайні елементи списку не враховуються, оскільки у них мало сусідів.

Вхідні дані В першому рядку записано число n. В наступному рядку записано n цілих чисел. Всі числа за модулем не перевищують 100.

Вихідні дані

Вивести кількість елементів, які більші своїх двох сусідів.

Складність: 9% — 2367/1109/1045/955.

Python (20 ms, 7.7 MiB)

```
n,k = int(input()),0
a = list(map(int,input().split()))
for i in range(1,n-1): k += a[i-1]<a[i]>a[i+1]
print(k)
```

Розв'язок є і в [16].

3.13.29. e7846 Найбільший елемент

Дано масив з n цілих чисел. Виведіть значення найбільшого елементу, а потім індекс цього елемента в масиві. Індексація елементів починається з 1. Якщо найбільших елементів декілька, виведіть індекс першого з них.

Вхідні дані В першому рядку записано число n. В наступному рядку записано n цілих чисел. Усі числа за модулем не перевищують 100.

Вихідні дані Вивести значення та індекс найбільшого елементу.

Input	**Output**
7 3 5 -7 7 5 -9 -4	7 4

Автор Матвійчук Сергій Володимирович

Джерело Серія задач "Абетка програмування"

Складність: 7% — 2468/1222/1169/1083.

Python (21 ms, 5.1 MiB)

```
input()
a=list(map(int,input().split()))
max=max(a)
print(max,a.index(max)+1)
```

Розв'язок є і в [16].

3.13.30. e7847 Кількість різних елементів

Дано масив з N цілих чисел. Визначте, скільки в цьому масиві різних елементів,

Вхідні дані В першому рядку записано число N. В наступному рядку записано N цілих чисел. Всі числа за модулем не перевищують 100.

Вихідні дані Кількість різних елементів в масиві.

Складність: 15% — 2196/881/853/725.

Використовуємо множину (елементів).

Python (35 ms, 7.4 MiB)

```
input()
print(len(set(input().split())))
```

Аналогічний розв'язок є в [16].

На **C++** [16]

```
#include <bits/stdc++.h>
using namespace std;
int main() {
        int n,t;
        set <int> S;
        cin >> n;
        for(int i=0; i<n; i++){
                cin >> t;
                S.insert(t);
        }
        cout << S.size();
}
```

3.13.31. e7848 Переставити сусідні

Дано масив з n цілих чисел. Переставте сусідні елементи масиву (a_0 з a_1, a_2 з a_3 і так далі). Якщо елементів непарна кількість, то останній елемент слід залишити на своєму місці.

Вхідні дані В першому рядку записано число n. В наступному рядку записано n цілих чисел. Всі числа за модулем не перевищують 100.

Вихідні дані Вивести оновлений масив.

Input	**Output**
7 3 5 -7 7 5 -9 -4	5 3 7 -7 -9 5 -4

Автор Матвійчук Сергій Володимирович

Джерело Серія задач "Абетка програмування"

Складність: 7% — 1844/1040/978/910.

Python (21 ms, 5.1 MiB)

```
n = int(input())
a = input().split()
for i in range(1,n,2): a[i],a[i-1]=a[i-1],a[i]
print(*a)
```

Розв'язок є і в [16].

C++ [25] (3 ms, 1.8 MiB)

```
#include <iostream>
using namespace std;
int main() {
```

```
  int n;
  cin >> n;
  int arr[n];
  for(int i = 0; i < n; i++) cin >> arr[i];
  for(int i = 1; i < n; i+=2){
    int q = arr[i];
    arr[i] = arr[i-1];
    arr[i-1] = q;
  }
  for(int i = 0; i < n; i++) cout << arr[i] << " ";
}
```

Напевно простіше буде без масивів **C++** (2 ms, 1.8 MiB)

```
#include <iostream>
using namespace std;
int main() {
  int n,f,s;
  cin >> n;
  for(int i = 0; i < n; i+=2){
    cin >> f >> s;
    cout << s << " " <<f << " ";
  }
  if(n%2) {
    cin >> f;
    cout << f;
  }
}
```

3.13.32. e7849 Обміняти max і min

Дано масив з n цілих чисел. Замінити всі найбільші елементи на найменший, а найменші елементи на найбільший.

Вхідні дані В першому рядку записано число n ($n \leq 100$). В наступному рядку записано n цілих чисел, кожне з яких за модулем не перевищує 100.

Вихідні дані Вивести оновлений масив.

Складність: 11% — 5065/2283/1977/1751.

Python

```
n = int(input())
```

```
a = list(map(int,input().split()))
min,max=min(a),max(a)
for i in range(n):
    if a[i]==min: a[i]=max
    if a[i]==max: a[i]=min
print(*a)
```

Розв'язок є і в [16].

3.13.33. e7850 Унікальні елементи

Дано масив з N цілих чисел. Виведіть ті його елементи, які зустрічаються в списку тільки один раз. Елементи потрібно виводити в тому порядку, в якому вони зустрічаються в списку.

Вхідні дані В першому рядку записано число N. В наступному рядку записано N цілих чисел. Всі числа за модулем не перевищують 100.

Вихідні дані Список унікальних елементів.

Складність: 13% — 1509/707/680/589.

Python (24 ms, 7.5 MiB)

```
input()
a=input().split()
for i in a:
    if a.count(i)==1: print(i,end='␣')
```

Розв'язок є і в [16].

На **C++** [16]

```
#include <bits/stdc++.h>
using namespace std;
int main() {
  int n,i,j,k,a[100];
  cin >> n;
  for(i=0; i<n; i++) cin >> a[i];
  for(i=0; i<n; i++) {
    k=0;
    for(j=0; j<n; j++)
      if (a[i]==a[j]) k++;
    if(k==1) cout << a[i] << "␣";
  }
}
```

3.13.34. e8959 Різниця між найбільшим і найменшим

Задано n цілих чисел. Вивести різницю між найбільшим і найменшим числом.

Вхідні дані В першому рядку записано число n ($1 \le n \le 100$). В другому рядку записано n цілих чисел, кожне з яких не перевищує за модулем 100.

Вихідні дані

Вивести різницю між найбільшим і найменшим числом.

Автор Матвійчук Сергій Володимирович

Джерело Серія задач "Абетка програмування"

Складність: 6% — 1718/962/890/836.

На **Python** (19 ms, 5.1 MiB)

```
input()
a=list(map(int,input().split()))
print(max(a)-min(a))
```

3.13.35. e8532 Друк квадратів і кубів

Задано два цілих числа a і b. Виведіть квадрати і куби всіх цілих чисел від a до b включно.

Вхідні дані Два цілих числа a і b ($0 \le a \le b \le 10000$).

Вихідні дані В першому рядку виведіть квадрати всіх цілих чисел від a до b включно за зростанням. В другому рядку виведіть куби всіх цілих чисел від a до b включно за спаданням.

Складність: 13% — 5348/1596/1463/1277.

Python (48 ms, 6.2 MiB)

```
a,b = map(int,input().split())
s,c = [],[]
for i in range(a,b+1): s.append(i**2); c.append(i**3)
print(*s)
print(*c[::-1])
```

На **C++** без масивів [25]

```
#include <iostream>
using namespace std;
int main() {
  int a, b;
```

```
  cin >> a >> b;
  for (long long int i = a; i <= b; i++)
    cout << i * i << ' ';
  cout << endl;
  for (long long int j = b; j >= a; j--)
    cout << j * j * j << ' ';
}
```

3.13.36. e0848 Досконалі числа

Число називається досконалим, якщо воно дорівнює сумі всіх своїх дільників, менших за нього. Потрібно знайти всі досконалі числа від M до N.

Вхідні дані У першому рядку знаходяться відокремлені пропуском числа M і N. M і N цілі; $1 \le M \le N \le 10^9$; $(N - M) \cdot Sqrt(N) \le 10^7$.

Вихідні дані

У кожному рядку вивести по одному числу у порядку зростання. Якщо досконалих чисел на проміжку немає, вивести "Absent".

Ліміт часу 5 с

Складність: 28% — 1753/397/439/315.

Оскільки досконалих чисел мало 2.1.1, заносимо їх в масив.

На **Python** (19 ms, 5.1 MiB)

```
p,f = [6,28,496,8128,33550336],1
m,n = map(int,input().split())
for i in range(5):
    if m<=p[i]<=n: print(p[i]); f=0
if f: print('Absent')
```

3.14. Обробка двовимірних масивів

3.14.1. e9560 Двовимірний масив — введення, виведення

Дано двовимірний масив розміром $n \times m$. Введіть двовимірний масив та виведіть його.

Вхідні дані В першому рядку вхідних даних записано два числа n та m, кількість рядків та кількість стовпців відповідно. Далі записа-

но n рядків по m чисел — елементи масиву. Всі числа за модулем не перевищують 100.

Вихідні дані Виведіть n рядків по m чисел — елементи масиву.

Input	**Output**
4 5 1 3 2 4 5 4 2 7 6 5 9 2 3 5 1 7 8 1 7 2	1 3 2 4 5 4 2 7 6 5 9 2 3 5 1 7 8 1 7 2

Автор Жуковський Сергій Станіславович
Джерело Серія задач "Абетка програмування"
Складність: 6% — 190/110/108/102.

Програма на **Python** (20 ms, 5.1 MiB)

```
n,m = map(int,input().split())
for i in range(n): print(input())
```

3.14.2. e9561 Найбільший в кожному стовпці

Дано двовимірний масив розміром $n \times m$. Для кожного стовпця знайти найбільший елемент.

Вхідні дані В першому рядку вхідних даних записано два числа n та m, кількість рядків та кількість стовпців відповідно. Далі записано n рядків по m чисел — елементи масиву. Всі числа за модулем не перевищують 100.

Вихідні дані В одному рядку вивести m чисел — відповідь до задачі.

Input	**Output**
4 5 1 3 2 4 5 4 2 7 6 5 9 2 3 5 1 7 8 1 7 2	9 8 7 7 5

Автор Жуковський Сергій Станіславович
Джерело Серія задач "Абетка програмування"
Складність: 14% — 133/61/66/57.

Програма на **Python** (30 ms, 5.2 MiB)

```
n,m, = map(int,input().split())
a,r=[],[]
for i in range(n):
```

```
    a.append(list(map(int,input().split())))
for j in range(m):
    mx = a[0][j]
    for i in range(1,n):
        if a[i][j]>mx: mx = a[i][j]
    r.append(mx)
print(*r)
```

3.14.3. e9562 Сума елементів підмасиву

Дано двовимірний масив розміром $n \times m$. Знайти суму елементів підмасиву.

Вхідні дані В першому рядку вхідних даних записано два числа n та m, кількість рядків та кількість стовпців відповідно. Далі записано n рядків по m чисел — елементи масиву. В останньому рядку записано чотири числа $r1, c1, r2, c2$ — номер рядка та номер стовпця лівого верхнього елемента та правого нижнього елемента підмасиву. ($1 \le n, m \le 100, r1 \le r2, c1 \le c2$). Всі числа за модулем не перевищують 100.

Вихідні дані Одне число — суму елементів підмасиву.

Input	Output
4 5 1 3 2 4 5 4 2 7 6 5 9 2 3 5 1 7 8 1 7 2 2 2 3 4	25

Автор Жуковський Сергій Станіславович

Джерело Серія задач "Абетка програмування"

Складність: 6% — 97/49/47/44.

Програма на **Python** (30 ms, 5.2 MiB)

```
n,m, = map(int,input().split())
a,s=[],0
for i in range(n):
    a.append(list(map(int,input().split())))
r1,c1,r2,c2 = map(int,input().split())
for i in range(r1-1,r2):
    for j in range(c1-1,c2): s += a[i][j]
print(s)
```

3.14.4. e9563 Рядки з мінімальними елементами

Дано двовимірний масив розміром $n \times m$. Знайдіть рядки, які місять мінімальний елемент.

Вхідні дані В першому рядку вхідних даних записано два числа n та m, кількість рядків та кількість стовпців відповідно. Далі записано n рядків по m чисел — елементи масиву. Всі числа за модулем не перевищують 100. Нумерація елементів масиву починається з 1.

Вихідні дані В одному рядку виведіть номери рядків, що містять мінімальний елемент, у зростаючому порядку.

Input	**Output**
4 5 2 5 3 1 4 3 5 2 3 4 4 6 1 2 3 4 5 6 7 7	1 3

Автор Жуковський Сергій Станіславович

Джерело Серія задач "Абетка програмування"

Складність: 20% — 422/118/129/103.

Вводимо (не формуючи двовимірний) масив і одразу знаходимо мінімум по рядкам, який і заносимо в масив (мінімумів) рядків. Шукаємо мінімум цього масиву. Проходимо по масиву і в масив результату заносимо положення (індекс+1) мінімуму.

Програма на **Python** (25 ms, 5.1 MiB)

```
n,m, = map(int,input().split())
r,R = [],[]
for i in range(n): r.append(min(map(int,input().split())))
mn = min(r)
for i in range(n):
    if r[i] == mn: R.append(i+1)
print(*R)
```

3.14.5. e9564 Рядки з максимальною сумою

Дано двовимірний масив розміром $n \times m$. Знайти рядки в яких сума елементів максимальна.

Вхідні дані В першому рядку вхідних даних записано два числа n та m, кількість рядків та кількість стовпців відповідно. Далі записано n рядків по m чисел — елементи масиву. Всі числа за модулем не перевищують 100. Нумерація елементів масиву починається з 1.

Вихідні дані В одному рядку виведіть номери рядків, в яких сума елементів максимальна.

Input	Output
4 5 2 5 3 1 4 3 5 2 3 4 4 6 1 2 3 4 5 6 7 7	1 3

Складність: 18% — 125/47/50/41.

Задача аналогічна попередній (3.14.4). Вводимо дані масиву і одразу знаходимо суму рядка, яку і заносимо в масив. Шукаємо максимум цього масиву. Проходимо по масиву і в масив результату заносимо положення (індекс+1) шуканих максимумів.

Програма на **Python** (23 ms, 5.1 MiB)

```
n,m, = map(int,input().split())
r,R = [],[]
for i in range(n): r.append(sum(map(int,input().split())))
mx= max(r)
for i in range(n):
    if r[i] == mx: R.append(i+1)
print(*R)
```

3.14.6. e9565 Мінімум серед максимумів

Дано двовимірний масив розміром $n \times m$. Знайти в кожному рядку максимальний елемент і серед максимальних елементів знайти мінімальний.

Вхідні дані В першому рядку вхідних даних записано два числа n та m, кількість рядків та кількість стовпців відповідно. Далі записано n рядків по m чисел — елементи масиву. Всі числа за модулем не перевищують 100.

Вихідні дані Виведіть одне число відповідь до задачі

Input	Output
4 5 1 5 3 2 4 2 4 3 2 2 6 7 8 9 5 8 9 4 2 5	4

Автор Жуковський Сергій Станіславович
Джерело Серія задач "Абетка програмування"
Складність: 15% — 97/50/53/45.

Задача схожа на попередні 3.14.4, 3.14.5. Вводимо масив і одразу знаходимо максимум по рядкам, який і заносимо в масив (мінімумів) рядків. Шукаємо мінімум цього масиву. Проходимо по масиву і в масив результату заносимо положення (індекс+1) мінімуму.

Програма на **Python** (25 ms, 5.1 MiB)

```
n,m, = map(int,input().split())
r,R = [],[]
for i in range(n): r.append(min(map(int,input().split())))
mn = min(r)
for i in range(n):
    if r[i] == mn: R.append(i+1)
print(*R)
```

3.14.7. e9566 Сортування по стовпцях

Дано двовимірний масив розміром $n \times m$. Відсортувати кожен стовпець у порядку спадання зверху до низу.

Вхідні дані В першому рядку вхідних даних записано два числа n та m, кількість рядків та кількість стовпців відповідно. Далі записано n рядків по m чисел — елементи масиву.

Вихідні дані Виведіть утворений масив

Input	**Output**
4 5	8 9 8 9 5
1 5 3 2 4	6 7 4 2 5
2 4 3 2 2	2 5 3 2 4
6 7 8 9 5	1 4 3 2 2
8 9 4 2 5	

Автор Жуковський Сергій Станіславович
Джерело Серія задач "Абетка програмування"
Складність: 23% — 122/52/60/46.

Вводимо двовимірний масив a. Перебираємо по стовпцях. Створюємо тимчасовий масив t і заносимо в нього поточний стовпець, сортуємо його і вертаємо в двовимірний масив.

Програма на **Python** (180 ms, 5.9 MiB)

```
n,m, = map(int,input().split())
a=[]
for i in range(n):
    a.append(list(map(int,input().split())))
for j in range(m):
    t=[]
    for i in range(n): t.append(a[i][j])
    t.sort(reverse=1)
    for i in range(n): a[i][j]=t[i]
for i in range(n): print(*a[i])
```

3.14.8. e9567 Зсунути нульові елементи праворуч

Дано двовимірний масив розміром $n \times m$. Зсуньте всі нульові елементи праворуч в кожному рядку.

Вхідні дані В першому рядку вхідних даних записано два числа n та m, кількість рядків та кількість стовпців відповідно. Далі записано n рядків по m чисел — елементи масиву.

Вихідні дані Виведіть утворений масив

Input	**Output**
4 5	1 5 2 4 0
1 5 0 2 4	4 2 0 0 0
0 4 0 2 0	6 9 5 0 0
6 0 0 9 5	8 9 4 0 0
8 9 4 0 0	

Автор Жуковський Сергій Станіславович

Джерело Серія задач "Абетка програмування"

Складність: 11% — 44/34/36/32.

Вводимо дані масиву і одразу обробляємо кожен рядок. Зсув нульових елементів проведемо аналогічно бульбашковому сортуванню, переставляємо в сусідній парі елементи, якщо лівий — нуль.

Програма на **Python** (22 ms, 5.1 MiB)

```
n,m, = map(int,input().split())
for i in range(n):
    a=list(map(int,input().split()))
    for k in range(-1,m-2):
        for j in range(m-2,k,-1):
            if a[j]==0: a[j],a[j+1]=a[j+1],a[j]
```

```
print(*a)
```

3.14.9. e9568 Зсунути нульові елементи вгору

Дано двовимірний масив розміром $n \times m$. Зсуньте всі нульові елементи вгору в кожному стовпці.

Вхідні дані В першому рядку вхідних даних записано два числа n та m, кількість рядків та кількість стовпців відповідно. Далі записано n рядків по m чисел — елементи масиву.

Вихідні дані Виведіть утворений масив

Автор Жуковський Сергій Станіславович

Джерело Серія задач "Абетка програмування"

Складність: 13% — 88/47/48/42.

Використовуємо, посилаємось на попередню 3.14.8. Вводимо двовимірний масив a. Перебираємо по стовпцях. Зсув нульових елементів проводимо аналогічно бульбашковому сортуванню, переставляючи в сусідній парі елементи, якщо нижній — нуль.

Програма на **Python** (22 ms, 5.1 MiB)

```
n,m, = map(int,input().split())
a=[]
for i in range(n):
    a.append(list(map(int,input().split())))
for j in range(m):
    for k in range(n,1,-1):
        for i in range(1,k):
            if a[i][j]==0:
                a[i][j],a[i-1][j]=a[i-1][j],a[i][j]
for i in range(n): print(*a[i])
```

3.14.10. e9569 Повернути масив за годинниковою стрілкою

Дано квадратний двовимірний масив розміром $n \times n$. Поверніть його елементи за часовою стрілкою.

Вхідні дані В першому рядку вхідних даних записано одне число n, розмір масиву.

Далі записано n рядків по n чисел — елементи масиву.

Вихідні дані Виведіть перетворений масив

Input	Output
4	4 3 2 1
1 1 1 1	4 3 2 1
2 2 2 2	4 3 2 1
3 3 3 3	4 3 2 1
4 4 4 4	

Автор Жуковський Сергій Станіславович
Джерело Серія задач "Абетка програмування"
Складність: 3% — 69/40/38/37.

Програма на **Python** (23 ms, 5.1 MiB)

```
n,a = int(input()),[]
for i in range(n): a.append(input().split())
r = [[0]*n for i in range(n)]
for i in range(n):
    for j in range(n): r[j][n-1-i] = a[i][j]
for i in range(n): print(*r[i])
```

3.14.11. e9570 Відобразити відносно вертикальної осі симетрії

Дано двовимірний масив розміром $n \times m$. Відобразіть його відносно вертикальної осі симетрії.

Вхідні дані В першому рядку вхідних даних записано два числа n та m, кількість рядків та кількість стовпців відповідно. Далі записано n рядків по m чисел — елементи масиву.

Вихідні дані Виведіть перетворений масив

Input	Output
2 3	3 2 1
1 2 3	6 5 4
4 5 6	

Автор Жуковський Сергій Станіславович
Джерело Серія задач "Абетка програмування"
Складність: 14% — 53/35/37/32.

Вводимо дані масиву порядково, перетворюємо в список і одразу виводимо опрацьований рядок.

Програма на **Python** (32 ms, 5.1 MiB)

```
n,m, = map(int,input().split())
for i in range(n):
    print(*(list(map(int,input().split())))[::-1])
```

3.14.12. e8941 Матриця

Задано два натуральних числа n і m. Вивести матрицю, що складається з n рядків та m стовпчиків, заповнену натуральними числами від 1 до $n * m$, як показано у прикладі.

Вхідні дані Два натуральних числа n та m.

Вихідні дані Вивести шукану матрицю.

Input	Output
2 3	1 2 3
	4 5 6

Автор Матвійчук Сергій Володимирович

Джерело Серія задач "Абетка програмування"

Складність: 8% — 1630/958/830/765.

Програма на **Python** (21 ms, 5.1 MiB)

```
n,m = map(int,input().split())
for i in range(n):
    print(*[j for j in range(m*i+1,m*(i+1)+1)])
```

3.15. Вектори

3.15.1. e2131 Довжина вектора

Обчислити довжину вектора.

Вхідні дані Чотири цілих числа x_1, y_1, x_2, y_2 — координати початку та кінця вектора відповідно. Усі вхідні числа не перевищують за модулем 10000.

Вихідні дані Одне число — довжина заданого вектора з точністю до шести десяткових знаків.

Складність: 5% — 6882/3805/3083/2936.

На **C#**

```
using System;
namespace e2131 {
```

```
  class Program      {
    static void Main(string[] args) {
      string[] z = Console.ReadLine().Split();
      Console.WriteLine("{0:f6}",
        Math.Sqrt(Math.Pow(Convert.ToInt32(z[2]) -
        Convert.ToInt32(z[0]), 2) +
        Math.Pow(Convert.ToInt32(z[3]) -
        Convert.ToInt32(z[1]), 2)));
    }
  }
}
```

На **Python**

```
x,y,X,Y=map(int,input().split())
print('%.6f'%((X-x)**2+(Y-y)**2)**.5)
```

3.15.2. e7449 Вектор. Скалярний добуток

Дано два вектори. Знайдіть їх скалярний добуток та кут між ними.

Вхідні дані Чотири цілих числа — координати ненульових векторів. Все числа за модулем не перевищують 10000.

Вихідні дані В першому рядку виведіть скалярний добуток двох векторів, а в другому виведіть величину неорієнтованого кута між векторами з точністю до п'яти десяткових знаків. Виведене число має належати інтервалу $[0; \pi]$.

Автор Михайло Медведєв

Складність: 11% — 395/189/166/147.

Python (43 ms, 5.5 MiB)

```
import math
x,y,X,Y=map(int,input().split())
sc = x*X+y*Y
print(sc)
print("%.5f"%math.acos((sc)/((x**2+y**2)*\
    (X**2+Y**2))**.5))
```

3.15.3. e2130 Кут між векторами

Обчислити кут між двома векторами.

Вхідні дані Чотири цілих числа — координати двох ненульових векторів. Усі вхідні числа не перевищують за модулем 10000.

Вихідні дані Вивести одне число — величину неорієнтовного кута між векторами з точністю 5 десяткових знаків. Число, що виводиться, має належати інтервалу $[0; \pi]$.

Складність: 13% — 5058/2101/1840/1592.

Задача є половиною (3.15.2).

В тестах контролюється кількість знаків після десяткової крапки.

На **Python** (26 ms, 5.2 MiB)

```
from math import acos
x,y,X,Y=map(int,input().split())
print('%.5f'%acos((x*X+y*Y)/((x**2+y**2)*  \
    (X**2+Y**2))**.5))
```

3.15.4. e2129 Полярний кут точки

Знайдіть полярний кут точки.

Вхідні дані Два цілих числа — декартові координати точки, яка не співпадає з початком координат. Вхідні числа не перевищують за модулем 10000.

Вихідні дані Вивести одне дійсне число — величину полярного кута вхідної точки в радіанах, що знаходиться в інтервалі $[0; 2\pi)$. Відповідь округлити з точністю до 6 знаків після десяткової коми.

Складність: 21% — 3360/862/903/714.

Для від'ємних кутів, перенесемо їх на 2π.

На **Python**

```
from math import atan2,pi
x,y=map(int,input().split())
print('%.6f'%((atan2(y,x)+2*pi)%(2*pi)))
```

3.15.5. e4776 Базові операції над вектором

Задано дві неспівпадаючі точки на площині, потрібно обчислити:

- Вектор с початком у першій і кінцем у другій точках
- Відповідний йому нормуючий вектор
- Вектор, співнаправлений з першим, який має задану довжину
- Вектор, отриманий шляхом повороту першого вектору на 90° за годинниковою стрілкою
- Вектор, отриманий шляхом повороту першого вектору на 90° проти годинникової стрілки

Вхідні дані У перших двох рядках задано по два цілих числа — координати заданих точок. У третьому рядку записано натуральне число — довжина, яку повинен мати побудований у третьому пункті задачі вектор. Усі числа у вхідному файлі по модулю не перевищують 1000.

Вихідні дані У окремих рядках вихідного файлу потрібно вивести координати векторів, що відповідають кожному з пунктів задачі, з точністю до 10^{-4}.

Складність: 23% — 496/236/154/118.

Python

```
b,B=map(int,input().split())
e,E=map(int,input().split())
x,y=e-b,E-B
print('%.9f %.9f'%(x,y))
l=(x**2+y**2)**.5
X,Y=x/l,y/l
print('%.9f %.9f'%(X,Y))
k=int(input())
print('%.9f %.9f'%(X*k,Y*k))
print('%.9f %.9f'%(y,-x))
print('%.9f %.9f'%(-y,x))
```

3.15.6. e4777 Вектори

Задано два ненульових вектори. Потрібно обчислити:
. Довжину першого та другого вектора (два числа)
. Вектор, утворений додаванням заданих двох векторів
. Скалярний та векторний добуток заданих векторів
. Площу трикутника, побудованого з цих векторів

Вхідні дані У двох рядках вхідного файлу задано по чотири цілих числа, які не перевищують по модулю 10000 — координати початку та кінця першого вектора, потім другого.

Вихідні дані У кожному рядку вихідного файлу - відповідь на відповідний пункт задачі з точністю не менше 10^{-6}.

Складність: 21% — 638/185/194/154.

Python

```
xb,yb,xe,ye=map(int,input().split())
Xb,Yb,Xe,Ye=map(int,input().split())
print(((xe-xb)**2+(ye-yb)**2)**.5, \
      ((Xe-Xb)**2+(Ye-Yb)**2)**.5)
print(xe-xb+Xe-Xb,ye-yb+Ye-Yb)
c=(xe-xb)*(Ye-Yb)-(Xe-Xb)*(ye-yb)
print((xe-xb)*(Xe-Xb)+(ye-yb)*(Ye-Yb),c)
print(abs(c)/2)
```

3.16. Матриці

3.16.1. e1482 Множення матриць

Нехай задано дві прямокутні матриці A та B розмірності $m \times n$ та $n \times q$ відповідно:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}, \; B = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1q} \\ b_{21} & b_{22} & \cdots & b_{2q} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{nq} \end{bmatrix}.$$

Тоді матриця C розмірності $m \times q$ називається їх добутком:

$$C = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1q} \\ c_{21} & c_{22} & \cdots & c_{2q} \\ \vdots & \vdots & \ddots & \vdots \\ c_{m1} & c_{m2} & \cdots & c_{mq} \end{bmatrix},$$

де $c_{i,j} = \sum_{i=1}^{n} a_{i,r} b_{r,j}$ $(i = 1, 2, \ldots m; \; j = 1, 2. \ldots q)$.

Операція множення двох матриць допустима лише у тому випадку, коли кількість стовпців у першому множнику дорівнює кількості рядків у другому; у цьому випадку кажуть, що форма матриць узгоджена.

Задано дві матриці A та B. Знайти їх добуток.

Вхідні дані

У першому рядку задано два натуральних числа n_a та m_a — розмірність матриці A. У наступних n_a рядках задано по m_a чисел — елемент a_{ij} матриці A. У (n_a+2)-му рядку задано два натуральних числа n_b та m_b — розмірність матриці B. У наступних n_b рядках задано по m_b чисел — елементи b_{ij} матриці B. Розмірність матриць не перевищує 100×100, усі елементи матриць цілі числа, які не перевищують за модулем 100.

Вихідні дані У першому рядку вивести розмірність результуючої матриці C: n_c та m_c. У наступних n_c рядках вивести через пропуск по m_c чисел — відповідні елементи c_{ij} матриці C. Якщо множити матриці не можна у першому і єдиному рядку вивести число -1.

Складність: 16% — 3666/1119/1017/853.

На **C++** (14 ms, 0.68 MiB)

```
#include <iostream>
using namespace std;
int main() {
  int na, ma;
  cin >> na >> ma;
  int a[100][100];
  for (int i = 0; i<na; i++)
    for (int j = 0; j<ma; j++) cin >> a[i][j];
  int nb, mb;
  cin >> nb >> mb;
  if (ma != nb) {cout << -1; return 0;}
  int b[100][100];
  for (int i = 0; i<nb; i++)
    for (int j = 0; j<mb; j++) cin >> b[i][j];
  int c[100][100];
  for (int i = 0; i<na; i++)
    for (int j = 0; j<mb; j++) {
      c[i][j] = 0;
      for(int k = 0; k<nb; k++)
        c[i][j]+=a[i][k]*b[k][j];
    }
  cout << na << " " << mb << endl;
  for (int i = 0; i<na; i++) {
    for (int j = 0; j<mb; j++) cout << c[i][j] << " ";
    cout << endl;
  }
}
```

На **Python** (483 ms, 6.9 MiB)

```
na,ma = map(int,input().split())
a,b,c=[],[],[]
for i in range(na):
    a.append([int(s) for s in input().split()])
nb,mb = map(int,input().split())
for i in range(nb):
    b.append([int(s) for s in input().split()])
if ma!=nb: print(-1); exit()
nc,mc=na,mb
print(nc,mc)
for i in range(nc):
    c.append([])
    for j in range(mc):
      c[i].append(0)
      for r in range(ma):
        c[i][j]+=a[i][r]*b[r][j]
for i in range(nc):
    for j in range(mc): c[i][j]=str(c[i][j])
for i in range(nc): print(*c[i])
```

3.17. Перебір

3.17.1. e0194 Добуток цифр

Знайти найменше і найбільше натуральні числа, добуток цифр у яких дорівнює заданому натуральному числу M або вивести -1 -1, якщо таких не існує. Для запису шуканих чисел не можна використовувати цифри 0 і 1.

У вхідному файлі ціле число M ($2 \le M \le 10^3$). До вихідного файлу потрібно записати два цілих числа в неспадному порядку.

Автор Сергій Матвійчук

Складність: 25% — 834/291/374/280.

Шукаємо спочатку мінімальне необхідне число. Перебираємо доступні цифри від 9 до 2. До тих пір поки число ділиться націло на поточну цифру ділимо число на неї та записуємо цифру на початку рядка відповіді.

Максимальне число шукаємо аналогічно, перебираючи цифри в порядку зростання.

Якщо при пошуку залишається число, що не ділиться націло на жодну цифру, виводимо -1 -1.

На **Python** (28 ms, 5.1 MiB)

```
m = M = int(input())
r = R = ''
while m>9:
    f=1
    for d in range(9,1,-1):
        if m%d==0: m//=d; r=str(d)+r; f=0; break
    if f: print(-1,-1); exit()
while M>3:
    for d in range(2,10):
        if M%d==0: M//=d; R=str(d)+R; break
print(str(m)+r,str(M)*(M>1)+R)
```

В [16] пропонується наступний підхід. Створюємо масиви цифр. Число розкладається на множники-цифри (2, 3, 5, 7) (неповна факторизація). Результат заноситься в масив цифр a, в якому запам'ятовуються кількості відповідних множників. Перелік символів з масиву b, що є копією a, в порядку спадання і є рядком максимального числа.

Для отримання рядка мінімального числа добутки 3×3, $2\times2\times2$, 2×3, 2×2 замінюємо на 9, 8, 6, 4, модифікуючи масив цифр.

Python

```
n,a,d = int(input()),[0]*10,2
while (n>1) and (d<8):
    if n%d==0: n //= d; a[d] += 1
    else: d += 1
b = a.copy()
while a[3]>1: a[9] += 1; a[3] -= 2
while a[2]>2: a[8] += 1; a[2] -= 3
while a[2]>0 and a[3]>0: a[6] += 1; a[2]-=1; a[3]-=1
while a[2]>1: a[4] += 1; a[2] -= 2
if n>1: print('-1 -1')
else:
    for i in range(1,10):
        for j in range(1,a[i]+1): print(i, end='')
    print(' ', end='')
    for i in range(9,0,-1):
        for j in range(1,b[i]+1): print(i, end='')
```

3.17.2. e0193 Сума цифр

Знайти найменше і найбільше N-значні натуральні числа, які мають суму цифр M.

У вхідному файлі числа N і M ($1 \le N \le 100,\ 1 \le M \le 9 \cdot N$). До вихідного файлу потрібно записати два N-значних числа в неспадному порядку.

Автор Сергій Матвійчук

Складність: 34% — 1300/368/545/359.

Перебираємо розряди n-розрядних чисел. Для кожного поточного розряду (крім останнього) цифра в мінімальному числі — максимальна цифра поточного мінімального числа (min(9,поточне мінімальне число -1), в максимальному — максимальну цифру (min(9,поточне максимальне число)). В діях одиницю віднімаємо оскільки вона може виявитись на початку мінімального n-цифрового числа. Після знаходження цифр розрядів зменшуємо на них відповідні числа. Знайдені цифри розрядів записуємо для мінімального числа на початку (результат формується вліво), а максимального — в кінці (результат формується вправо). Останні шукані цифри — залишок попередніх дій.

На **Python** (32 ms, 5.5 MiB)

```
n, m,   = map(int, input().split())
r,R,d,D = '','',m,m
for j in range(n - 1):
    q = min(9,d-1); w = min(9,D)
    r = str(q) + r; R += str(w)
    d -= q;         D -= w
print(str(d)+r, R+str(D))
```

Деякий розв'язок є в [16].

3.17.3. e9648 Сортування цифр числа

Розглянемо послідовність всіх натуральних чисел від a до b. В кожному числі відсортуємо цифри по зростанню. Знайдіть суму отриманих чисел.

Вхідні дані Два натуральних числа a і b ($a \le b \le 10^6$).

Вихідні дані Виведіть суму отриманих чисел.

Приклад Нехай $a = 19, b = 22$. Числами от a до b будуть: $19, 20, 21, 22$. Після сортування цифр в числах отримуємо числа: $19, 02, 12, 22$. Сума чисел рівна $19 + 2 + 12 + 22 = 55$.

Ліміт часу 2 с

Автор Михайло Медведєв

Складність: 25% — 19/9/8/6.

Звертаємо увагу на збільшений ліміт часу. Повільний Python дає вихід за цей ліміт.

Переберемо числа в послідовності. Для кожного числа розкладемо його по цифрам в масив d. Сортуємо отриманий масив цифр бульбашковим сортуванням.

В масиві t накопичуємо суми цифр в розрядах чисел.

Потрібну суму збираємо з масиву t за схемою Горнера.

Програма на **C++** (46 ms, 1.8 MiB)

```
#include <iostream>
using namespace std;
int main() {
    int a,b;
    cin >> a >> b;
    long long s=0, t[7]={0};
    for(int n=a; n<=b; n++){
        int d[7]={0};
        int k=0, i=n;
        while(i){d[k]=i%10; i /= 10; k++;}
        for(int i=0; i<k-1; i++){
            for(int j=0; j<k-i-1; j++){
                if(d[j+1]>d[j]){
                    int tmp=d[j+1];
                    d[j+1]=d[j];
                    d[j]=tmp;
                }
            }
        }
        for(int j=0; j<7; j++) t[j] += d[j];
    }
    for(int j=6; j>=0; j--) s = s*10+t[j];
    cout << s << endl;
}
```

3.17.4. e0140 Фінансова піраміда

В понеділок Толя позичив у Сергійка 2 цукерки і з задоволенням їх з'їв. У вівторок він позичив у два рази більше цукерок, після чого віддав половину боргу, а решту цукерок знову із задоволенням з'їв. Кожний наступний день він позичав у два рази більше цукерок, ніж у попередній день, віддавши з них цілу частину від половини боргу, а решту цукерок із задоволенням з'їдав. Скільки цукерок K із задоволенням з'їсть Толя N-го дня? Який у нього буде борг B на кінець N-го дня ?

Вхідні дані У вхідному файлі одне число N. $1 \le N \le 30$.

Вихідні дані У вихідний файл потрібно записати два числа — значення K та B.

Автор Сергій Матвійчук

Джерело II етап Всеукраїнської олімпіади з інформатики в Житомирській обл.

Складність: 24% — 2544/764/943/719.

Python (32 ms, 5.5 MiB)

```
n,k,b=int(input()),2,2
for i in range(2,n+1):
    o=2**i
    v=(o+b)//2
    k=o-v
    b+=k
print(k,b)
```

Аналогічний розв'язок є і в [16].

3.17.5. e0542 Постачання содової води

Тім дуже полюбляє содову воду, інколи він нею ніяк не може напитися. Ще більш прикрим є той факт, що у нього постійно бракує грошей. Тому єдиним легальним способом їх отримання є продаж порожніх пляшок з-під соди. Іноді на додаток до його особисто випитих пляшок додаються ті, які Тім іноді знаходить на вулиці. Одного дня Тіма настільки замучила спрага, що він вирішив пити до тих пір поки міг собі це дозволити.

Вхідні дані Три цілі невід'ємні числа e, f, c, де e ($e < 1000$) — кількість порожніх пляшок, які є у Тіма на початку дня, f ($f < 1000$) — кількість порожніх пляшок, знайдених протягом дня, і c ($1 < c < 2000$) — кількість порожніх пляшок, необхідних для покупки нової пляшки.

Вихідні дані Скільки пляшок содової води зможе випити Тім, коли його замучила спрага?

Джерело

2009 Nordic Collegiate Programming Contest, Жовтень 3, Задача A

Складність: 11% — 7301/3431/2978/2652.

Змоделюємо процес купівлі/випивання, здачі пляшок і т.д.

На **Python** (31 ms, 5.4 MiB)

```
e,f,c=map(int,input().split())
e += f
n = 0
while e >= c: e -= c-1; n += 1
print(n)
```

3.17.6. e0016 Дракон

У кожної S-ніжки 1 голова. Знайти кількість ніг N у K-голового дракона, якщо разом у всіх A голів і B ніг.

Вхідні дані 4 числа: S, K, A, B. Всі числа не перевищують 1000.

Вихідні дані Кількість ніг у дракона. Якщо вхідні дані суперечні, вивести у вихідний файл -1, у випадку наявності декількох розв'язків — вивести довільний з них.

Складність: 37% — 10364/1641/2202/1381.

На **Pascal**

```
var S, K, A, B, N, rez:longint;
begin
   readln (S, K, A, B);
   if s*a=b then writeln (s*k)
      else
        begin
          n:=1;
          while n<1001 do
             begin
                if (abs(s*a-b)) mod n =0 then
                   begin
                      rez:=((b-s*a) div n) +s*k;
```

```
                    if (rez>=0) and (a>n*k) then
                      begin
                        writeln (rez);
                        halt;
                      end;
                  end;
                inc (n);
              end;
            if n=1001 then writeln (-1);
          end;
end.
```

На **Python**

```
s,k,a,b=map(int,input().split())
r=-1
if(s*a==b):r=s*k
else:
  for n in range(s*k):
    if(((s*a-b)%(s*k-n)==0) & ((s*a-b)*(s*k-n)>0) & \
        ((n-k*s)*(n*a-b*k)>0)):
      r=n
      break
print(r)
```

Інші задачі з перебором

e0134 Два кола – 2 Умова та розв'язок наведені в 3.20.37.

3.18. Рекурсія

3.18.1. e0849 Розклад на доданки

Вивести всі подання натурального числа N сумою натуральних чисел. Перестановка доданків нового способу подання не дає.

Вхідні дані

У першому рядку знаходиться єдине число N. $2 \le N \le 40$.

Вихідні дані У кожному рядку виводиться один зі способів подання. У поданні суми доданки відокремлюються знаком «+».

Складність: 15% — /3270/3823/2732.

Рекурсивно перебираємо.

На **Pascal** (34 ms, 0.68 MiB)

```
var a:array[1..40] of integer;
    n:integer;
procedure recurse(step,start,left:integer);
  var i,j:integer;
  begin
    if start>left then exit;
    if step>1 then
      begin
        a[step]:=left;
        write(a[1]); for j:=2 to step do
          write('+',a[j]); writeln;
      end;
    for i:=start to left do
      begin
        a[step]:=i;
        recurse(step+1,i,left-i);
      end;
  end;
begin
  readln(n);
  recurse(1,1,n);
end.
```

3.19. Жадібний алгоритм

3.19.1. e8788 Монети

У Вас є нескінчена кількість монет номіналами від 1 до n. Ви хочете вибрати певний набір монет сумою s. Дозволено мати в наборі монети з однаковим номіналом. Яку мінімальну кількість монет необхідно взяти, щоб набрати суму s.

Вхідні дані Два цілих чисел n та s ($1 \le n \le 10^5,\ 1 \le s \le 10^9$).

Вихідні дані Виведіть мінімальну кількість монет, необхідну для взяття суми s.

Складність: 9% — 176/94/94/86.

На **Python** (50 ms, 5.1 MiB)

```
n,s = map(int,input().split())
k=0
for i in range(n,0,-1):
    k += s//i
    s %= i
    if s==0: break
print(k)
```

3.19.2. e0138 Банкомат

Банкомат містить в достатній кількості купюри номіналом 10, 20, 50, 100, 200 і 500 гривень. Знайти найменшу кількість купюр, якою можна видати суму в n гривень, або вивести -1, якщо вказану суму видати не можна.

Вхідні дані Одне число n ($1 \le n \le 1000000$).

Вихідні дані

Найменша кількість купюр, якою можна видати n гривень.

Автор Сергій Матвійчук

Джерело II етап Всеукраїнської олімпіади з інформатики в Житомирській обл.

Складність: 14% — 16544/6165/5986/5151.

Жадібний алгоритм

Pascal

```
const d:array[0..5] of integer =(500,200,100,50,20,10);
var n,i,k:longint;
begin
  read(n);
  for i:=0 to 5 do
    begin
      k:=k+n div d[i];
      n:=n mod d[i];
    end;
  if n>0 then k:=-1;
  writeln(k)
end.
```

Python

```
d,n,k= [500,200,100,50,20,10],int(input()),0
for i in d: k+=n//i; n%=i
if n: k=-1
print(k)
```

Інші задачі на жадібний алгоритм

e0007 Римські числа (113), **e4103** Римські числа (112).

3.20. Геометричні задачі

Система координат, точка

3.20.1. e0918 Яка чверть?

Задано точку з координатами **x** та **y**. Визначити, в якій координатній чверті вона розміщена.

Вхідні дані У єдиному рядку через пропуск задано 2 дійсні числа — координати точки, значення координат по модулю не перевищують 100.

Вихідні дані Єдине число — номер відповідної чверті, або 0, якщо однозначно визначити чверть неможливо.

Ліміт часу 0.5 с

Джерело ДПА-2010 Варіант 18

Складність: 16% — 32154/8567/8634/7265.

Pascal

```
var x,y:real;
begin
  readln(x,y);
  if (x=0) or (y=0) then Writeln(0) else
  begin
    if x > 0 then
      if y > 0 then Writeln(1) else Writeln(4)
    else
      if y > 0 then writeln(2) else Writeln(3);
  end;
end.
```

Найкращим розв'язком є використання умов (як чисел), враховуючи, що в 1-й та 3-й чвертях x і y є однакового знаку, 3-й та 4-й — $y < 0$.

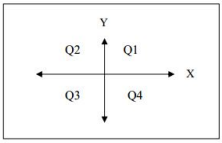
Pascal

```
var x,y:real;
begin
  readln(x,y);
  Writeln(ord(x*y<>0)*(ord(x*y<0)+2*ord(y<0)+ 1));
end.
```

Python

```
x,y = map(float,input().split())
print((x*y != 0)*((x*y < 0) + 2*(y < 0) + 1))
```

3.20.2. e6938 Квадранти



Для координат (x, y) деяких точок у двовимірній площині, з'ясуйте, якому квадрату (Q1-Q4) належать точки. Деякі точки належать AXIS, якщо вони розташовані на осі X або Y.

Вхідні дані Перший рядок містить ціле число n ($1 \le n \le 1000$), яке визначає кількість точок. Наступні n рядків містять два цілих числа, які відповідають координатам (x_i, y_i) кожної точки ($-10^6 \le x_i, y_i \le 10^6$).

Вихідні дані Виведіть загальну кількість точок у кожному квадранті та осі у тому ж форматі, що і у наведеному нижче зразку.

Input	**Output**
5	Q1: 2
0 0	Q2: 0
0 1	Q3: 0
1 1	Q4: 1
3 -3	AXIS: 2
2 2	

Джерело ACM-ICPC Asia Phuket Regional Programming Contest 2013, Practice Session, 21 November 2013

Складність: 1% — 275/186/160/159.

Використовуємо, спираємось на 3.20.1.

Python (37 ms, 5.5 MiB)

```
n,q,a=int(input()),[0,0,0,0],0
for i in range(n):
    x,y=map(int,input().split())
    if x*y==0: a+=1
    else: q[(x*y<0)+2*(y<0)]+=1
for i in range(4): print('Q',i+1,': ',q[i],sep='')
print('AXIS:',a)
```

Пряма, промінь, відрізок

3.20.3. e2141 Рівняння прямої I

Знайдіть коефіцієнти загального рівняння прямої.

Вхідні дані Чотири числа — координати двох різних точок на прямій. Усі вхідні дані цілі числа, які не перевищують по модулю 10000.

Вихідні дані Три цілих числа — коефіцієнти **A, B** і **C** загального рівняння цієї прямої.

Складність: 6% — 1237/745/697/653.

Програма на **Python** (35 ms, 5,5 MiB)

```
x,y,X,Y=map(int,input().split())
A,B=Y-y,x-X
print(A,B,-A*x-B*y)
```

3.20.4. e2142 Рівняння прямої II

Знайдіть коефіцієнти загального рівняння прямої.

Вхідні дані Чотири числа — координати точки на прямій і координати вектора нормалі до цієї прямої. Усі координати цілі числа, які не перевищують по модулю 10000.

Вихідні дані Три цілих числа — коефіцієнти **A, B** і **C** загального рівняння цієї прямої.

Складність: 14% — 649/337/361/309.

Програма на **Python** (35 ms, 5,5 MiB)

```
x,y,X,Y=map(int,input().split())
A,B=Y-y,x-X
print(A,B,-A*x-B*y)
```

3.20.5. e2132 Належність точки прямій

Визначте, чи належить точка прямій, яка задана рівнянням $Ax + By + C = 0$.

Вхідні дані П'ять цілих чисел — координати точки x, y та коефіцієнти A, B, C рівняння прямої (гарантується, що A та B одночасно не дорівнюють 0).

Вихідні дані Вивести «YES», якщо точка належить прямій і «NO» у протилежному випадку.

Input	Output
3 7 -2 1 -1	YES

Складність: 4% — 4111/2680/2305/2210.

Якщо точка лежить на прямій, її координати задовільняють рівнянню.

Python

```
x,y,a,b,c = map(int,input().split())
print('YES') if a*x+b*y+c==0 else print('NO')
```

3.20.6. e2133 Належність точки променю

Визначте, чи належить задана точка променю.

Вхідні дані Містить шість цілих чисел — координати точки та координати початку та кінця вектора. Усі числа не перевищують за модулем 10000.

Вихідні дані Вивести **YES**, якщо точка належить променю та **NO** у протилежному випадку.

Складність: 33% — 4989/913/1103/744.

Python

```
x,Y,b,B,e,E=map(int,input().split())
print('YES') if (b-x)*(E-B)-(e-b)*(B-Y)==0 and \
   (b-x)*(e-b)+(B-Y)*(E-B)<=0 else print('NO')
```

3.20.7. e2136 Відстань від точки до прямої

Знайдіть відстань від заданої точки до заданої прямої.

Вхідні дані Шість цілих чисел — координати точки і координати двох точок, якими задано пряму. Вхідні дані не перевищують по модулю 10000.

Вихідні дані
Одне число — відстань від точки до прямої з точністю 10^{-6}.
Складність: 12% — 1565/687/690/607.

Програма на **Python** (28 ms, 5,5 MiB)

```
x,y,x1,y1,x2,y2=map(int,input().split())
dx,dy=x2-x1,y2-y1
print('%.6f'%(abs((x-x1)*dy-(y-y1)*dx)/(dx**2+dy**2)**.5))
```

3.20.8. e2144 Відстань від точки до прямої

Знайдіть відстань від заданої точки до заданої прямої.
Вхідні дані П'ять цілих чисел — координати точки і коефіцієнти A, B і C нормального рівняння прямої. Усі вхідні дані цілі числа, які не перевищують по модулю 10000.
Вихідні дані Одне число — відстань від точки до прямої з точністю не менше 10^{-6}.
Складність: 8% — 1390/573/559/514.

Програма на **Python** (38 ms, 5,5 MiB)

```
x,y,A,B,C=map(int,input().split())
print("%.6f"%(abs(A*x+B*y+C)/(A**2+B**2)**.5))
```

3.20.9. e2137 Відстань від точки до променя

Знайдіть відстань від заданої точки до заданого променя.
Вхідні дані Шість цілих чисел, які не перевищують по модулю 10000, — координати точки і координати початку і кінця вектора.
Вихідні дані Одне число — відстань від точки до променя, заданого вектором, з точністю 10^{-6}.
Складність: 30% — 1431/274/338/236.

Програма на **Python** (34 ms, 5,5 MiB)

```
X,Y,x,y,xe,ye=map(int,input().split())
dX,dY,dx,dy=X-x,Y-y,xe-x,ye-y
print('%.6f'%(abs(dX*dy-dx*dY)/(dx**2+dy**2)**.5))  \
  if dx*dX+dy*dY>0 else print('%.6f'%((dX**2+dY**2)**.5))
```

3.20.10. e2143 Перетин двох прямих

Знайти координати точки перетину двох непаралельних прямих.

Вхідні дані Шість чисел — коефіцієнти A, B і C нормального рівняння двох різних непаралельних прямих (спочатку для однієї прямої, потім для другої). Усі вхідні дані цілі числа, які не перевищують по модулю 10000.

Вихідні дані Два числа – координати точки їх перетину з точністю 2 знаки після коми.

Складність: 44% — 2339/396/424/236.

Програма на **Python** (31 ms, 5,4 MiB)

```
a,b,c,A,B,C=map(int,input().split())
d=A*b-a*B
print("%.2f %.2f"%((B*c-b*C)/d,(a*C-A*c)/d))
```

3.20.11. e1353 Відрізок в системі координат

Дано відрізок з координатами кінців x_1, y_1, x_2, y_2. В яких координатних чвертях лежить відрізок?

Вхідні дані В одному рядку записано 4 цілих числа x_1, y_1, x_2, y_2 ($-50 \le x_1, y_1, x_2, y_2 \le 50$).

Вихідні дані

Виведіть у порядку зростання номери чвертей, у яких лежить відрізок. Кожне число необхідно вивести з нового рядка, або 0 якщо відрізок лежить на координатній вісі.

Складність: 77% — 3238/174/543/127.

Використовуємо, спираємось на ??e0918).

Python (35 ms, 5.5 MiB)

```
def Q(x,y):
    if x*y!=0: q.add((x*y<0)+2*(y<0))
x,y,X,Y=map(int,input().split())
if x==X==0 or y==Y==0: print(0); exit()
q=set()
Q(x,y)
Q(X,Y)
if X-x!=0 and Y-y!=0 and x*X<=0 and y*Y<=0:
    C=x*Y-y*X
    Q(C/(Y-y),C/(x-X))
```

```
for i in sorted(list(q)): print(i+1)
```

3.20.12. e0938 Точка на відрізку

Відрізок задано координатами його кінців $M(x_1, y_1)$, $N(x_2, y_2)$. Знайти координати точки $O(x, y)$, що ділить його у відношенні α.

Вхідні дані В одному рядку задано координати кінців відрізка та число α. Усі координати не перевищують за модулем 100.

Вихідні дані В одному рядку вивести координати x та y шуканої точки з точністю до сотих.

Складність: 10% — 5000/2577/2470/2218.

Pascal

```
var xm,ym,xn,yn,a:real;
begin
  read(xm,ym,xn,yn,a);
  writeln((xn*a+xm)/(a+1):0:2,' ',
          (yn*a+ym)/(a+1):0:2)
end.
```

Python

```
b,B,e,E,k=map(float,input().split())
print('%.2f %.2f'%((b+k*e)/(k+1),(B+k*E)/(k+1)))
```

3.20.13. e2134 Належність точки відрізку

Визначити, чи належить задана точка відрізку.

Вхідні дані Шість цілих чисел — координати точки і координати початку і кінця відрізку. Усі числа не перевищують за модулем 10000.

Вихідні дані Вивести «YES», якщо точка належить відрізку, і «NO» у протилежному випадку.

Складність: 37% — 5809/924/1176/738.

Програма на **Python** (28 ms, 5,5 MiB)

```
x,y,x1,y1,x2,y2=map(int,input().split())
x1,x2,y1,y2=min(x1,x2),max(x1,x2),min(y1,y2),max(y1,y2)
print('YES') if (x2==x1==x and y1<=y<=y2) or \
  (y==y1==y2 and x1<=x<=x2) or \
```

```
((x-x1)*(y2-y1)==(y-y1)*(x2-x1) and x1<=x<=x2 and \
y1<=y<=y2) else print('NO')
```

3.20.14. e4778 Належність точки проміжку

Визначте, чи належить точка **C** заданій прямій, променю та відрізку, утвореним точками **A** та **B**.

Вхідні дані У першому рядку вхідного файлу задано два цілих числа — координати точки **C**. У двох наступних рядках у такому ж форматі задано точки **A** та **B** ($A \neq B$). Усі числа у вхідному файлі по модулю не перевищують 10000.

Вихідні дані У першому рядку виведіть **YES**, якщо точка **C** належить прямій **AB**, і **NO** у протилежному випадку. У другому та третьому рядках аналогічно виведіть відповіді для променя **AB** (**A** — початок променя) та відрізка **AB**.

Складність: 43% — 848/83/136/78.

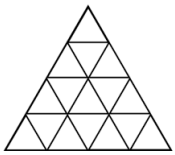
Програма на **Python** (28 ms, 5,5 MiB)

```
c,C=map(int,input().split())
a,A=map(int,input().split())
b,B=map(int,input().split())
f = (a-b)*(A-C)-(A-B)*(a-c)==0
print('YES') if f else print('NO')
print('YES') if f and (b-a)*(c-a)+(B-A)*(C-A)>=0 else \
  print('NO')
print('YES') if f and (a-c)*(b-c)+(A-C)*(B-C)<=0 else \
  print('NO')
```

Трикутник

3.20.15. e2400 Трикутники

Михайлик любив малювати трикутники, але він це робив у незвичний спосіб. Спочатку малював довільний трикутник, потім кожну сторону ділив на n рівних частин і проводив через точки поділу прямі, паралельні сторонам трикутника. У результаті виходить декілька рівних між собою трикутників.



Допоможіть Михайлику знайти найбільшу кількість однакових трикутників у його фінальному рисунку.

Вхідні дані Ціле число n ($0 < n < 2 \cdot 10^9$).

Вихідні дані

Вивести найбільшу кількість рівних між собою трикутників.

Складність: 21% — 4760/1925/1895/1506.

Найбільшою кількістю рівних між собою трикутників може бути тільки кількість найменших тут трикутників.

Кількість вказаних трикутників зростає між сусідніми рядками трикутників на 2, тобто утворює арифметичну прогресію (2.1.3) з $a_1 = 1$, $d = 2$. Сума цієї прогресії $\frac{(2 \cdot 1 + 2(n-1))}{2} n = n^2$.

На **Python**

```
print(int(input())**2)
```

Також розв'язок є в [16].

В мовах з статичною типізацією потрібно використовувати 64-бітні типи (long long, longint, int64, ...).

На **C#**

```
using System;
class tr {
  private static void Main() {
    ulong a = Convert.ToUInt64(Console.ReadLine());
        Console.WriteLine(a*a);
  }
}
```

3.20.16. e0915 Прямокутний чи ні?

Задано довжини сторін трикутника. Визначити, чи є цей трикутник прямокутним.

Вхідні дані У єдиному рядку задано 3 натуральні числа — довжини сторін трикутника. Довжини сторін не перевищують 1000.

Вихідні дані Вивести "YES"(без лапок), якщо трикутник є прямокутним, або "NO"(без лапок) у протилежному випадку.

Джерело ДПА-2010 Варіант 15

Складність: 11% — 21044/8527/8081/7208.

Скористуємось теоремою Піфагора 3.20.16. Для визначення гіпотенузи відсортуємо введені числа.

Програма на **Pascal**

```
var a,b,c,t:int64;
begin
    read(a,b,c);
    if a>c then begin t:=a; a:=c; c:=t end;
    if b>c then begin t:=b; b:=c; c:=t end;
    if a*a+b*b=c*c then writeln('YES')
                   else writeln('NO')
end.
```

Програма на **C#**

```
using System;
namespace e0915 {
  class Program {
    private static void Main(string[] args) {
      string[] a = Console.ReadLine().Split('␣');
      int[] n = new int[3];
      for (int i = 0; i < 3; i++)
        n[i] =Convert.ToInt32(a[i]);
      Array.Sort(n);
      if (Math.Pow(n[2],2) == Math.Pow(n[0],2)+
          Math.Pow(n[1],2))
          Console.WriteLine("YES");
      else
          Console.WriteLine("NO");
    }
  }
}
```

Програма на **Python**

```
a=sorted(list(map(int,input().split())))
print('YES') if(a[0]**2+a[1]**2==a[2]**2) else \
  print('NO')
```

Розв'язок на **C++** [16]

```
#include <bits/stdc++.h>
```

```
using namespace std;
int main() {
  int a,b,c;
  cin >> a >> b >> c;
  if(a*a+b*b==c*c || b*b+c*c==a*a || c*c+a*a==b*b)
    cout << "YES" << endl;
  else
    cout << "NO" << endl;
}
```

3.20.17. e0905 Який трикутник?

Визначити тип трикутника (рівносторонній, рівнобедрений, різносторонній) за заданими довжинами його сторін.

Вхідні дані В одному рядку задано 3 цілих числа — довжини сторін трикутника. Довжини сторін не перевищують 100.

Вихідні дані Вивести **1**, якщо трикутник рівносторонній, **2** — якщо рівнобедрений і **3** — якщо різносторонній.

Складність: 9% — 29027/12392/10955/9936.

Pascal

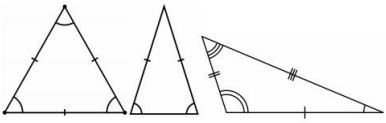
```
var a,b,c,r:byte;
begin
  read(a,b,c);
  r:=3;
  if (a=b)or(a=c)or(b=c) then r:=2;
  if (a=b)and(b=c) then r:=1;
  writeln(r)
end.
```

Python (22 ms, 5.1 MiB)

```
print(len(set(map(int,input().split()))))
```

Деякі розв'язки є в [16], [26](C++).

3.20.18. e2732 Трикутник



Трикутник — одна з основних фігур геометрії: багатокутник з трьома кутами або вершинами і трьома сторонами або ребрами, які є відрізками ліній.

Трикутники можна класифікувати за відносною довжиною їх сторін:

- У рівносторонньому трикутнику всі сторони мають однакову довжину. Рівносторонній трикутник — це також регулярний многокутник з усіма кутами 60°.
- У рівнобедреному трикутнику дві сторони рівні по довжині. Трикутник рівнобедреного типу також має два однакових кути; а саме кути, протилежні двом сторонам однакової довжини; цей факт є змістом теореми рівнобедреного трикутника.
- У масштабному трикутнику всі сторони неоднакові. Три кути і також всі по мірі різні. Деякі (але не всі) масштабні трикутники також є правильними трикутниками.

Вхідні дані

Перший рядок введення містить ціле число ($1 \leq T \leq 100$), кількість тестових випадків. Далі ідуть T тестових наборів даних, кожен складається з 3 цілих чисел A, B і C, де ($1 \leq A, B, C \leq 1000000$) довжини сторін трикутника.

Вихідні дані

Для кожного тестового випадку виведіть «equilateral», «isosceles» або «scalene», що описує тип трикутника. Якщо введення не створює допустимий вихід трикутника «invalid!». Дотримуйтесь наведеного нижче формату.

Input	**Output**
2	Case #1: isosceles
3 3 4	Case #2: invalid!
6 4 2	

Джерело The Third Lebanese Collegiate Programming Contest

Складність: 17% — 1434/432/475/396.

Програма на **Python** (27 ms, 5,5 MiB)

```
t,k=int(input()),0
for i in range(t):
    k+=1
    print('Case␣#',k,':␣',sep='',end='')
    a,b,c=map(int,input().split())
```

```
if a+b<=c or a+c<=b or b+c<=a: print('invalid!')
elif a==b==c: print('equilateral')
elif a==b or a==c or b==c: print('isosceles')
else: print('scalene')
```

3.20.19. e0925 Периметр та площа трикутника

Задано дійсні числа $x_1, y_1, x_2, y_2, x_3, y_3$, значення яких відповідають координатам вершин трикутника. Визначити периметр та площу трикутника.

Вхідні дані У єдиному рядку через пропуск задано координати вершин трикутника: 6 чисел $x_1, y_1, x_2, y_2, x_3, y_3$, значення яких не перевищують за модулем 100.

Вихідні дані В єдиному рядку вивести периметр та площу трикутника, обчислену з точністю до 4-х знаків після десяткової крапки.

Складність: 10% — 12897/5920/5037/4544.

Використаємо формулу Герона 3.21.7

Python (35 ms, 5.2 MiB)

```
def l(X,Y,x,y): return ((X-x)**2+(Y-y)**2)**.5
x1,y1,x2,y2,x3,y3 = map(float, input().split())
a,b,c = l(x1,y1,x2,y2),l(x1,y1,x3,y3),l(x2,y2,x3,y3)
p=(a+b+c)/2
print(2*p,(p*(p-a)*(p-b)*(p-c))**.5)
```

3.20.20. e0666 Трикутник і точка

Визначіть, чи лежить задана точка всередині заданого трикутника.

Вхідні дані Перші 3 рядки містять координати вершин трикутника (у кожному рядку по 2 цілих числа, відокремлених пропуском). Четвертий рядок містить координати точки, у такому ж форматі. Всі числа — цілі, по модулю не перевищують 10000. Гарантується, що вершини трикутника не лежать на одній прямій.

Вихідні дані Єдиний рядок містить слово «In», якщо точка лежить всередині трикутника, «On», якщо точка лежить на границі трикутника (вершині або стороні), або «Out», якщо вона лежить поза ним.

Складність: 36% — 6734/1044/1293/823.

Для точки в середині трикутника сума площ трикутників з вершинами в заданій точці дорівнює площі заданого трикутника, точка за межами трикутника — сума більше.

Оскільки у вхідних даних тільки цілі числа, **не проводимо розрахунки** за їх множиною.

Порівнювати будемо подвійні площі. Ці площі рахуємо за векторним добутком сторін трикутників як векторів.

На **Pascal**

```
var ax,ay,bx,by,cx,cy,x,y:integer;
    s1,s2,s3,s0:longint;
function s(ax,ay,bx,by,cx,cy:integer):longint;
  begin
    ax:=ax-cx; ay:=ay-cy;
    bx:=bx-cx; by:=by-cy;
    s:=abs(ax*by-ay*bx);
  end;
begin
  readln(ax,ay);
  readln(bx,by);
  readln(cx,cy);
  readln(x ,y );
  s0:=s(ax,ay,bx,by,cx,cy);
  s1:=s(ax,ay,bx,by,x ,y );
  s2:=s(ax,ay,cx,cy,x ,y );
  s3:=s(bx,by,cx,cy,x ,y );
  if s0<(s1+s2+s3) then writeln('Out') else
    if s1*s2*s3=0 then writeln('On')
                  else writeln('In')
end.
```

На **Python**

```
x1,y1=map(int,input().split())
x2,y2=map(int,input().split())
x3,y3=map(int,input().split())
x,y=map(int,input().split())
s=abs((x2-x1)*(y3-y1)-(y2-y1)*(x3-x1))
s1=abs((x1-x)*(y2-y)-(y1-y)*(x2-x))
s2=abs((x1-x)*(y3-y)-(y1-y)*(x3-x))
s3=abs((x2-x)*(y3-y)-(y2-y)*(x3-x))
```

```
if s<s1+s2+s3: print('Out')
elif s1*s2*s3==0: print('On')
else: print('In')
```

3.20.21. e0932 Висота трикутника

Визначити висоту трикутника площею S, якщо його основа більша за висоту на величину a.

Вхідні дані Два цілих числа S ($0 < S \leq 100$) та a ($|a| \leq 100$).

Вихідні дані Вивести висоту трикутника з точністю до сотих.

Складність: 10% — 8046/3671/3393/3038.

C#

```
using System;
namespace e0932 {
  class Program {
    static void Main(string[] args) {
      string[] input = Console.ReadLine().Split(' ');
      double s = Convert.ToInt16(input[0])*2;
      double a = Convert.ToDouble(input[1])/2;
      Console.WriteLine("{0:0.00}",Math.Sqrt(a*a+s)-a);
    }
  }
}
```

Python

```
S,a = map(int,input().split())
print('%.2f'%(((a/2)**2+2*S)**.5-a/2))
```

3.20.22. e0934 Висоти трикутника

Обчислити висоти трикутника зі сторонами a, b, c.

Вхідні дані У єдиному рядку через пропуск три натуральні числа — сторони трикутника: a, b, c. Всі вхідні дані не перевищують 100.

Вихідні дані Висоти, опущені до відповідних сторін через пропуск: h_a, h_b, h_c. Результат вивести з точністю 2 цифри після десяткової крапки.

Input	Output

3 4 5	4.00 3.00 2.40

Складність: 8% — 8578/4113/3650/3369.

Python

```
a,b,c = map(int, input().split())
p = (a+b+c)/2
s = 2*(p*(p-a)*(p-b)*(p-c))**.5
print('%.2f %.2f %.2f'%(s/a,s/b,s/c))
```

3.20.23. e0418 Трикутник

Дано трикутник і точку у ньому. Через цю точку проведено прямі, паралельні сторонам трикутника. Ці прямі утворюють три трикутника всередині даного. Їх площі відомі. Знайти площу заданого трикутника.

Вхідні дані Один рядок містить три додатні дійсні числа **S1, S2** та **S3**. Всі числа не перевищують 1000.

Вихідні дані Вивести площу заданого трикутника. Відповідь виводити з 8 десятковими знаками.

Складність: 6% — 1219/825/749/705.

Використовуємо подібність трикутників.

Python (22 ms, 5.5 MiB)

```
s1,s2,s3=map(float,input().split())
print('%.8f'%(((s1)**.5+(s2)**.5+(s3)**.5)**2))
```

3.20.24. e5186 Центр вписаного кола

Знайдіть координати центру вписаного кола для заданого трикутника.

Вхідні дані Координати трьох вершин трикутника (спочатку координати першої вершини, потім другої, потім третьої). Координати — пара цілих чисел, які не перевищують 10^4 за модулем.

Вихідні дані Виведіть два числа — координати центра вписаного кола з точністю не менше 6 десяткових знаків.

Складність: 14% — 225/82/90/77.

Програма на **Python** (23 ms, 7.8 MiB)

```
xa,ya=map(int,input().split())
xb,yb=map(int,input().split())
xc,yc=map(int,input().split())
a=((xb-xc)**2+(yb-yc)**2)**.5
b=((xa-xc)**2+(ya-yc)**2)**.5
c=((xb-xa)**2+(yb-ya)**2)**.5
print((xa*a+xb*b+xc*c)/(a+b+c),(ya*a+yb*b+yc*c)/(a+b+c))
```

3.20.25. e1614 Кути трикутника

Задано трикутник. Визначте величину самого великого з його кутів.

Вхідні дані Координати трьох вершин трикутника (спочатку координати першої вершини, потім другої, потім третьої). Координатами є пара цілих чисел, що не перевищують 10^4 за модулем.

Вихідні дані Виведіть величину самого великого кута трикутника у градусах з 6 десятковими знаками.

Складність: 22% — 2629/874/806/629.

Скористаємось теоремою косинусів. Використаємо масив x — координат вершин трикутника та масив a — квадратів його сторін. Всі розрахунки проведемо в цілих числах, за винятком самої останньої операції — добування квадратного кореня, знаходження арккосинусу та ділення на π.

На **Python** (27 ms, 5.2 MiB)

```
from math import acos,pi
x,a=[],[]
for i in range(3):
  x.append(list(map(int,input().split())))
x.append(x[0])
for i in range(3):
  a.append((x[i][0]-x[i+1][0])**2+(x[i][1]-x[i+1][1])\
    **2)
a.sort()
print('%.6f'%(acos((a[0]+a[1]-a[2])/2/(a[0]*a[1])**.5)\
    *180/pi))
```

Чотирикутник

3.20.26. e0130 Прямокутник

Дано координати трьох точок, вершин прямокутника. Знайдіть координати четвертої точки.

Вхідні дані В єдиному рядку записано шість чисел координати трьох точок.

Вихідні дані Два числа, координати шуканої вершини прямокутника. Всі вхідні та вихідні дані — цілі числа по модулю не перевищують 100.

Джерело II етап Всеукраїнської олімпіади з інформатики в Житомирській обл.

Складність: 45% — 6110/1199/1980/1087.

В протилежних вершинах Δx, Δy однакові.

Python (28 ms, 5.5 MiB)

```
x1,y1,x2,y2,x3,y3 = map(int,input().split())
if    (x2-x1)*(x3-x1)+(y2-y1)*(y3-y1)==0:
        print(x2+x3-x1,y2+y3-y1)
elif  (x3-x2)*(x1-x2)+(y3-y2)*(y1-y2)==0:
        print(x3+x1-x2,y3+y1-y2)
else: print(x1+x2-x3,y1+y2-y3)
```

3.20.27. e0133 Квадрат і точки

Яку найбільшу кількість точок з цілочисельними координатами можна на аркуші в клітинку накрити квадратом зі стороною N клітинок?

Вхідні дані Єдине число — сторона квадрату N ($1 \le N \le 10000$).

Вихідні дані Максимальна кількість накритих клітин K.

Складність: 10% — 12647/6108/5633/5052.

Можна показати, що кількість шуканих точок не залежить орієнтації квадрату на площині. Для цього можна розрізати квадрат горизонтальними та вертикальними лініями та перемістити отримані трикутники так щоб отримати квадрат з горизонтальними та вертикальними сторонами і тою ж кількість точок з цілочисельними координатами. Перемістимо квадрат так, щоб сторони мали одну з цілочисельних координат. Тоді кількість точок буде максимальною. На кожній з сторін буде розташовано $N+1$ точку, разом $(N+1)^2$ точок в квадраті.

На **Pascal**

```
var n:integer;
begin
    read(n);
    writeln((n+1)*(n+1))
end.
```

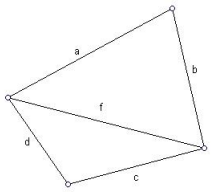
На **C#**

```
using System;
class Program{
  static void Main(string[] args){
    int n=Convert.ToInt32(Console.ReadLine());
    Console.WriteLine((n + 1) * (n + 1));
  }
}
```

На **Pascal** (20 ms, 5.1 MiB)

```
print((int(input())+1)**2)
```

3.20.28. e0926 Формула Герона



Задано сторони a, b, c, d та діагональ f опуклого чотирикутника. Визначити площу чотирикутника, використовуючи допоміжну функцію обчислення площі трикутника за формулою Герона.

Вхідні дані В одному рядку задано 5 дійсних чисел a, b, c, d, f $(0 < a, b, c, d, f \leq 100)$, як це показано на рисунку.

Вихідні дані Вивести площу чотирикутника, обчислену з точністю до 4-х знаків після десяткової крапки.

Джерело ДПА-2010 Варіант 26

Складність: 12% — 15465/6472/5708/5039.

Формула Герона (2.6.1)

На **Pascal**

```
var a,b,c,d,f:real;
function s(a,b,c:real):real;
```

```
  var p:real;
  begin
    p:=(a+b+c)/2;
    s:=Sqrt(p*(p-a)*(p-b)*(p-c))
  end;
begin
  read(a,b,c,d,f);
  writeln(s(a,b,f)+s(c,d,f):0:4)
end.
```

На **Python**

```
def s(a,b,c):
    p = (a+b+c)/2
    return (p*(p-a)*(p-b)*(p-c))**.5
a,b,c,d,f=map(float, input().split())
print(round(s(a,b,f)+s(c,d,f),4))
```

3.20.29. e0962 Найбільша сторона чотирикутника

На площині задано чотирикутник координатами своїх вершин. Обчислити довжину найбільшої сторони чотирикутника.

Вхідні дані У єдиному рядку через пропуск координати **X** та **Y** вершин чотирикутника: спочатку точки **A**, потім **B**, далі **C** і **D**. Усі вхідні дані цілі числа, які не перевищують по модулю 100.

Вихідні дані Єдине число — довжина найбільшої сторони. Результат вивести з точністю до сотих.

Складність: 16% — 3152/1230/1332/1117.

На **Python** (29 ms, 5.4 MiB)

```
a,A,b,B,c,C,d,D = map(int,input().split())
print('%.2f'%(max((a-b)**2+(A-B)**2,(b-c)**2+(B-C)**2,\
    (c-d)**2+(C-D)**2,(d-a)**2+(D-A)**2))**.5)
```

3.20.30. e1359 Сторона квадрата

Знайти цілочисельну довжину сторони квадрата, який можна отримати з двох прямокутників $a \times b$ та $c \times d$, розрізавши їх на прямокутники, а потім склавши так, щоб утворився квадрат найбільш можливої площі.

Вхідні дані У одному рядку знаходиться чотири дійсних числа a, b, c, d. Площа кожного прямокутника не перевищує 2×10^9.

Вихідні дані Вивести одне число — сторону утвореного квадрата.

Складність: 21% — 1334/446/503/398.

Площа отриманого квадрату не перевищує сумарну площу прямокутників.

Python (24 ms, 5.1 MiB)

```
a,b,c,d = map(int,input().split())
print(int((a*b+c*d)**.5))
```

Розв'язок є і в [16].

3.20.31. e0769 Прямокутник

Петрику потрібно вибрати на площині 4 точки так, щоб вони утворювали прямокутник зі сторонами, паралельними осям координат. Петрик вже вибрав три точки і впевнений, що він вибрав їх вірно. Допоможіть Петрику знайти координати четвертої точки.

Вхідні дані Містить три рядки. Кожен рядок містить два натуральних числа, відокремлених пропуском —– координати однієї з вершин прямокутника. Всі координати лежать у діапазоні від 1 до 1000.

Вихідні дані Вивести два цілих числа — координати четвертої вершини прямокутника.

Складність: 9% — 2204/1283/1195/1091.

З трьох координат x чи y потрібно вибрати координату, яка не збігається з парою інших однакових.

Найцікавішим розв'язком [16] тут є використання **виключно або** (3.1.7) — $a \hat{} a = 0$, $0 \hat{} b = b$. Таким чином $a \hat{} b \hat{} c$ дає число, що відрізняється від пари двох однакових (або збігається з ними, якщо всі три однакові, що не зустрічається для заданого прямокутника).

C++

```
#include <iostream>
using namespace std;
int main() {
    int x1,x2,x3,y1,y2,y3;
    cin >> x1 >> y1 >> x2 >> y2 >> x3 >>y3;
    cout << (x1^x2^x3) << ' ' << (y1^y2^y3);
}
```

Python

```
x1,y1=map(int,input().split())
x2,y2=map(int,input().split())
x3,y3=map(int,input().split())
print(x1^x2^x3,y1^y2^y3)
```

3.20.32. e0144 Чотирикутник

Довільний чотирикутник на площині заданий послідовно координатами своїх вершин. Визначити кількість прямих кутів чотирикутника.

Вхідні дані

Складається з послідовності координати вершин чотирикутника, значення яких цілочисельні та за модулем не перевищують 100.

Вихідні дані Вивести кількість прямих кутів чотирикутника.

Складність: 20% — 6486/2436/2475/1992.

Для прямого кута скалярний добуток векторів (3.15.2), на яких побудовано чотирикутник дорівнює нулю. Не виходимо за межі цілих чисел.

Для спрощення програми всі координати розташовуємо в одному масиві, з циклічним продовженням (альтернативно додаємо ще раз координати першої точки в кінці масиву).

Python

```
x,k=[int(s) for s in input().split()],0
x.extend(x)
for i in range(0,7,2):
    if (x[i]-x[i+2])*(x[i+2]-x[i+4])+(x[i+1]-x[i+3])* \
       (x[i+3]-x[i+5])==0: k+=1
print(k)
```

3.20.33. e0929 Паралелограм

Задано 4 числа a, b, c, d, що визначають довжини відрізків. Визначити, чи можна з цих відрізків утворити паралелограм.

Вхідні дані У єдиному рядку задано 4 числа через пропуск.

Вихідні дані Вивести у єдиному рядку слово «YES», якщо паралелограм утворти можна або «NO» (без лапок) у протилежному випадку.

Складність: 17% — 19000/5771/5962/4961.

C#

```
using System;
namespace e0929 {
  class Program {
    static void Main(string[] args) {
      string[] v=Console.ReadLine().Split('␣');
      Array.Sort(v);
      if (v[0]==v[1] && v[2]==v[3])
          Console.WriteLine("YES");
      else
          Console.WriteLine("NO");
    }
  }
}
```

Python

```
v = sorted(input().split())
print('YES') if v[0]==v[1] and v[2]==v[3] else \
 print('NO')
```

Многокутник

3.20.34. e0060 Площа многокутника

Задано координати n послідовних вершин многокутника. Знайти його площу.

Вхідні дані Перший рядок містить кількість вершин многокутника n ($3 \le n \le 50000$). У наступних n рядках задано цілочисельні координати його послідовних вершин x_i, y_i ($-1000 \le x_i, y_i \le 1000$).

Вихідні дані

Вивести площу многокутника з трьома десятковими знаками.

Складність: 21% — 8060/2594/2813/.

Оскільки координати цілочисельні, рахуємо подвійну площу многокутника як суму подвійних площ трикутників, користуючись векторним добутком (як в 3.15.6, 3.20.20).

Python (22 ms, 5.5 MiB)

```
n=int(input())
x1,y1=map(int,input().split())
```

```
x,y,S=x1,y1,0
for i in range(n-1):
    x2,y2=map(int,input().split())
    S += (y2+y1)*(x2-x1)
    x1,y1=x2,y2
S += (y+y2)*(x-x2)
print('%.3f'%(abs(S)/2))
```

3.20.35. e7333 Паркан

Емо переїхали в недавно побудовані будинки і вирішили побудувати паркан навколо свого поселення. Емо дивні люди, вони будуть плакати, якщо принаймні одна секція забору не паралельна осі координат. Тому Ваше завдання — побудувати паркан мінімальної довжини навколо поселення Емо таким чином, щоб всі будинки розташовувалися всередині області, обмеженої забором. Паркан повинен являти собою багатокутник без самоперетинів і самодотикань зі сторонами, паралельними осям координат. Будинки є точки з заданими координатами. Деякі будинки можуть знаходитися на самій огорожі.

Вхідні дані Перший рядок містить кількість будинків n ($2 \le n \le 100\,000$). Наступні n рядків містять опис будинків — кожен рядок містить два цілих числа x_i, $y_i (-10^9 \le x_i, y_i \le 10^9)$. Всі будинки розташовані в різних точках, як мінімум дві x-координати і y-координати різні.

Вихідні дані

Вивести одне ціле число — найменшу можливу довжину паркану.

Джерело

2014 ACM-ICPC Україна, 2-й Раунд, Вересень 13, Задача J

Складність: 18% — 452/160/169/138.

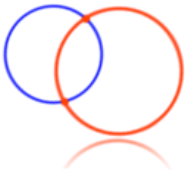
Проекція паркану на кожну вісь не залежить від форми описаного паркану і дорівнює різниці координат по цій осі.

Python (273 ms, 13.1 MiB)

```
n,x,y=int(input()),[],[]
for i in range(n):
    X,Y=map(int,input().split())
    x.append(X)
    y.append(Y)
print(2*((max(x)-min(x)+(max(y)-min(y)))))
```

Коло, круг, кільце

3.20.36. e0004 Два кола



Визначити в скількох точках перетинаються два кола.

Вхідні дані 6 чисел $x_1, y_1, r_1, x_2, y_2, r_2$, де x_1, y_1, x_2, y_2, — координати центрів кіл, r_1, r_2 — їх радіуси. Всі числа — дійсні, не перевищують 1 000 000 000 за модулем, та задані не більш ніж з 3 знаками після коми.

Вихідні дані Кількість точок перетину. Якщо точок перетину нескінченно багато, то вивести -1.

Складність: 39% — 58423/8266/10906/6698.

Програма на **C#**

```
using System;
using System.Collections.Generic;
using System.Linq;
namespace e0004 {
  class Program {
    static void Main(string[] args) {
      string input = Console.ReadLine();
      double x = Convert.ToDouble(input.Split(' ')[0]);
      double y = Convert.ToDouble(input.Split(' ')[1]);
      double r = Convert.ToDouble(input.Split(' ')[2]);
      double X = Convert.ToDouble(input.Split(' ')[3]);
      double Y = Convert.ToDouble(input.Split(' ')[4]);
      double R = Convert.ToDouble(input.Split(' ')[5]);
      double d = Math.Sqrt(Math.Pow(x - X, 2) +
        Math.Pow(y - Y, 2));
      if (x == X && y == Y && r == R)
          Console.WriteLine(-1);
      else if (d > r + R || d < Math.Abs(R - r))
          Console.WriteLine(0);
      else if (d == r+R || d == Math.Abs(R - r))
          Console.WriteLine(1);
      else
          Console.WriteLine(2);
    }
  }
```

```
}
```

Програма на **Python**

```
x,y,r,X,Y,R = map(float,input().split())
l=((X-x)**2+(Y-y)**2)**.5
if x==X and y==Y and r==R: print(-1)
elif l>r+R or l<abs(r-R): print(0)
elif l==r+R or l==abs(r-R): print(1)
else: print(2)
```

Розв'язок на C++ є в [16].

3.20.37. e0134 Два кола – 2

На площині побудовано 2 кола, відповідно з центрами у точках $O_1(X_1, Y_1)$ та $O_2(X_2, Y_2)$ і радіусами R_1 та R_2.

Скільки різних точок з цілочисельними координатами міститься у двох колах?

Вхідні дані Координати центра та радіуси кіл: X_1, Y_1, R_1, X_2, Y_2, R_2. Всі вхідні дані цілі числа, що не перевищують за модулем 100.

Вихідні дані Шукана кількість точок.

Складність: 17% — 1135/483/570/473.

Переберемо всі точки з цілочисельними координатами з області, що займають кола. Працюємо тільки з цілими числами. Порівнюємо квадрати відстаней від центрів кіл до заданої точки з квадратами радіусів.

Програма на **Python** (409 ms, 5,5 MiB)

```
x,y,r,X,Y,R=map(int,input().split())
xm,ym,XM,YM,k=min(x-r,X-R),min(y-r,Y-R),max(x+r,X+R), \
    max(y+r,Y+R),0
for i in range(xm,XM+1):
    for j in range(ym,YM+1):
        if ((i-x)**2+(j-y)**2<=r**2) or \
            ((i-X)**2+(j-Y)**2<=R**2): k+=1
print(k)
```

3.20.38. e3171 Точка всередині круга

Перевірити, чи знаходиться точка всередині круга.

Вхідні дані У першому рядку задано координати центра круга та його радіус. У другому рядку задано координати точки **A**. Усі числа цілі, не перевищують за модулем 10000.

Вихідні дані Вивести «YES», якщо точка **A** належить кругу (з границями), і «NO» інакше.

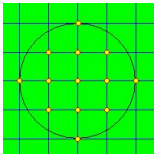
Складність: 9% — 2720/1232/1124/1021.

Працюємо тільки з цілими числами.

Python (35 ms, 5.5 MiB)

```
X,Y,R=map(int,input().split())
x,y=map(int,input().split())
print('NO') if (X-x)**2+(Y-y)**2>R**2 else print('YES')
```

3.20.39. e0295 Круг



Скільки точок з цілочисельними координатами знаходиться у крузі радіусом r? Точка, що знаходиться на колі, вважається належною кругу. Центр кола має цілочисельні координати.

Вхідні дані Цілочисельний радіус кола r ($r \le 15\,000$).

Вихідні дані Вивести шукану кількість точок.

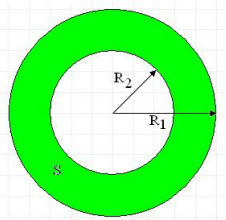
Складність: 37% — 4591/1034/1413/886.

Розташуємо центр кола в початку координат. Розглянемо одну чверть. В координаті x знаходиться $\lfloor\sqrt{r^2 - x^2}\rfloor$ цілочисельних координат $y \neq 0$. Додамо точку в початку координат.

Python (28 ms, 5.1 MiB)

```
r,k = int(input()),0
for x in range(r): k += int((r**2-x**2)**.5)
print(4*k+1)
```

3.20.40. e0924 Кільце



Задано площу кільця й радіус зовнішнього кола. Визначити радіус внутрішнього кола.

Вхідні дані Задано два дійсних числа: площа кільця і радіус зовнішнього кола, шо не перевищує 100.

Вихідні дані Вивести радіус внутрішнього кола з 2 десятковими знаками.

Джерело ДПА-2010 Варіант 24

Складність: 13% — 18761/6497/6107/5335.

Площа кільця $S = \pi(R^2 - r^2)$ **??**. Тому потрібний радіус $r = \sqrt{R^2 - \frac{S}{\pi}}$.

На **Python** (29 ms, 5.4 MiB)

```
from math import sqrt, pi
S, R = map(float, input().split())
print('%.2f' % sqrt(R**2 - S/pi))
```

3d фігури

3.20.41. e0944 Площа піраміди

Трикутна піраміда задана координатами своїх вершин $A(x_1; y_1; z_1)$, $B(x_2; y_2; z_2)$, $C(x_3; y_3; z_3)$, $S(x_4; y_4; z_4)$. Визначити площу повної поверхні піраміди.

Вхідні дані У чотирьох рядках задані координати x, y та z вершин піраміди. Усі числа цілі, за модулем не перевищують 100.

Вихідні дані

Вивести повну поверхню піраміди з точністю до десятих.

Складність: 9% — 2537/1373/1305/1191.

Площу кожної грані рахуємо за формулою Герона **??**

Python

```
def s(a,b,c):
    p=(a+b+c)/2
    return (p*(p-a)*(p-b)*(p-c))**.5
ax,ay,az=map(int,input().split())
bx,by,bz=map(int,input().split())
cx,cy,cz=map(int,input().split())
sx,sy,sz=map(int,input().split())
ab=((ax-bx)**2+(ay-by)**2+(az-bz)**2)**.5
ac=((ax-cx)**2+(ay-cy)**2+(az-cz)**2)**.5
bc=((cx-bx)**2+(cy-by)**2+(cz-bz)**2)**.5
sa=((ax-sx)**2+(ay-sy)**2+(az-sz)**2)**.5
sb=((sx-bx)**2+(sy-by)**2+(sz-bz)**2)**.5
```

```
sc=((sx-cx)**2+(sy-cy)**2+(sz-cz)**2)**.5
print('%.1f'%(s(ab,ac,bc)+s(ac,sa,sc)+ \
    s(ab,sa,sb)+s(sb,sc,bc)))
```

3.20.42. e0948 Площа та об'єм піраміди

Сторона основи правильної чотирикутної піраміди d, бічне ребро p. Визначити площу повної поверхні та об'єм піраміди.

Вхідні дані У єдиному рядку через пропуск основа та бічне ребро. Вхідні дані не перевищують 100.

Вихідні дані Через пропуск шукані площа та об'єм, результат вивести з точністю до тисячних.

Input	Output
20 15	847.214 666.667

Складність: 10% — 3128/1444/1464/1312.

З прикладу видно, що вхідні дані є цілими.

Python

```
using System;
namespace e0948 {
  class Program {
    static void Main(string[] args) {
      string[] input = Console.ReadLine().Split(' ');
      double d = Convert.ToDouble(input[0]);
      double p = Convert.ToDouble(input[1]);
      double a = Math.Sqrt(p*p-d*d/4);
      double h = Math.Sqrt(p*p-d*d/2);
      double S = d*(d+2*a);
      double V = d*d*h/3;
      Console.WriteLine("{0:0.000} {1:f3}",S,V);
    }
  }
}
```

Python

```
d,p=map(int,input().split())
a,h=(p**2-d**2/4)**.5,(p**2-d**2/2)**.5
print('%.3f %.3f'%(d*(d+2*a),d*d*h/3))
```

3.20.43. e0889 Циліндр

Із аркушу паперу ножицями Ви можете вирізати дві поверхні, з яких можна скласти циліндр наступним чином:

1. Розрізати папір горизонтально (паралельно короткій стороні), отримавши дві прямокутні частини.
2. З першої частини вирізати круг максимального радіуса. Він буде лежати в основі циліндра.
3. Скрутіть другу прямокутну частину у трубочку так щоб її периметр дорівнював довжині кола, що обмежує круг. Прикріпіть трубочку до основи циліндра. Відмітимо, що трубочка може містити накриваючу частину паперу, так як її радіус підганяли до довжини радіуса основи циліндра.

За заданими розмірами паперу необхідно побудувати подібним чином циліндр максимального об'єму.

Вхідні дані Вхідні дані містять декілька тестів. Кожен тест містить два числа w і h ($1 \le w \le h \le 100$), які позначають ширину та висоту аркуша паперу. Останні тест містить два нулі і не опрацьовується.

Вихідні дані

Для кожного тесту у окремому рядку вивести значення найбільшого можливого об'єму циліндру, який можна побудувати з аркушу паперу заданих розмірів. Об'єм слід виводити з 3 десятковими знаками.

Складність: 39% — 157/42/61/37.

На **Python** (93 ms, 5.5 MiB)

```
from math import pi
while 1:
    w,h=map(float,input().split())
    if w+h==0: break
    d=w/pi
    V=d**2*pi/4*(h-d)
    v=min(h/(pi+1),w)**2*pi/4*w
    print("%.3f"%(max(V,v)))
```

3.21. Дати та час

3.21.1. e0147 Кількість днів

Знайти кількість днів між двома календарними датами, причому початковий і кінцевий дні враховуються також.

Вхідні дані В першому і другому рядку вхідного файлу записано по одній календарній даті у форматі $D/M\ Y$ (день місяць рік). $1 \le D \le 31,\ 1 \le M \le 12,\ 1 \le Y \le 2100$.

Вихідні дані У вихідний файл потрібно записати одне число — кількість днів між датами.

Автор Сергій Матвійчук

Джерело II етап Всеукраїнської олімпіади з інформатики в Житомирській обл.

Складність: 51% — 2431/324/629/307.

Програма на **Python**

```
import datetime
d,m,y = map(int,input().split())
D,M,Y = map(int,input().split())
print(abs((datetime.date(Y,M,D)-datetime.date(y,m,d)).\
    days)+1)
```

3.21.2. e0125 Олімпіада

Олімпіада почалася в $h1$ год $m1$ хв $s1$ с, а закінчилася цієї ж календарної доби в $h2$ год $m2$ хв $s2$ с. Скільки часу (год хв сек) тривала олімпіада?

Вхідні дані У першому рядку записано час початку, а у другому — час закінчення олімпіади у форматі **год хв сек** ($0 \le h1 \le h2 \le 23, 0 \le m1, m2 \le 59, 0 \le s1, s2 \le 59$).

Вихідні дані

Вивести час, який тривала олімпіада у форматі **год хв сек**.

Джерело II етап Всеукраїнської олімпіади з інформатики в Житомирській обл.

Складність: 19% — 18369/6212/6405/5184.

Python

```
h,m,s=map(int,input().split())
H,M,S=map(int,input().split())
```

```
ds = S-s
dm = M-m+ds//60
print(H-h+dm//60, dm%60, ds%60)
```

3.21.3. e6279 Кількість днів у місяці

Вивести кількість днів в N–му місяці M–го року по григоріанському календарю.

Історичний факт: *Наприкінці 1582 року папа Григорій XIII3.21.7 запровадив у католицьких країнах календар, у якому рік є високосним, якщо він кратний 4, але не кратний 100, або ж кратний 400.*

Вхідні дані Значення N та M ($1 \le N \le 12$, $1 \le M \le 2100$).

Вихідні дані Знайдена кількість днів.

Складність: 29% — 4359/1014/1175/836.

На **Python** (29 ms, 5.5 MiB)

```
n=[0,31,28,31,30,31,30,31,31,30,31,30,31]
m,Y = map(int,input().split())
md = n[m]
if m==2 and (Y%4==0 and Y%100!=0 or Y%400==0): md+=1
print(md)
```

Деякий розв'язок є і в [16].

3.21.4. e7226 День календаря

Як відомо день програміста припадає на 256 день року, у невисокосний рік це — 13 вересня, а у високосний — 12. Не забудьте привітати своїх колег і наставників.

Аналогічно пропонується розпізнати число та номер місяця, що припадає на день за номером n у невисокосному 2014 році.

Вхідні дані Натуральне число n ($1 \le n \le 365$).

Вихідні дані Число (від 1 до 31) та номер місяця (від 1 до 12), що відповідає дню з номером n.

Складність: 12% — 1535/576/546/479.

На **Python** (29 ms, 5.5 MiB)

```
d,n,m=[0,31,59,90,120,151,181,212,243,273,304,334,365],\
  int(input()),1
while n>d[m]: m+=1
```

print(n–d[m–1],m)

Деякий розв'язок є і в [16].

3.21.5. e6602 Години і хвилини

Хайді має дискретний аналоговий годинник у формі кола, як той, що на малюнку. Дві стрілки обертаються навколо центру кола, позначаючи години та хвилини. На годиннику розміщено 60 міток по всьому периметру, при цьому відстань між послідовними знаками є постійною. Хвилинна стрілка рухається від її поточної позначки до наступної рівно один раз на хвилину. Часова стрілка переміщується від її поточної позначки до наступної рівно один раз кожні 12 хвилин, тому вона просувається на п'ять знаків щогодини. Ми вважаємо, що обидві стрілки рухаються дискретно і миттєво, це означає, що вони завжди розташовані точно над однією з позначок і ніколи не знаходяться між знаками.

Опівночі обидві стрілки досягають одночасно верхньої позначки, яка вказує на нуль годин та нуль хвилин. Через рівно 12 годин або 720 хвилин обидві стрілки знову досягають одного і того ж положення, і цей процес повторюється знову і знову. Зверніть увагу, що при русі хвилини стрілки годинна стрілка може не рухатися; однак, коли годинна рухається, хвилинна стрілка також рухається.

Хайді подобається геометрія, і їй подобається вимірювати мінімальний кут між двома стрілками годинника в різний час дня. Вона записувала деякі заходи, але через кілька років і довгий список, вона помітила, що деякі кути повторюються, а інші ніколи не з'являються. Наприклад, у списку Хайді вказується, що і в три години, і в дев'яту годину мінімальний кут між двома стрілками становить 90 градусів, а кут 65 градусів у цьому списку не відображається. Хайді вирішила перевірити, чи існує будь-яке ціле число A від 0 до 180, чи існує принаймні один момент дня, щоб мінімальний кут між двома стрілками годинника був рівно A градусів. Допоможіть їй програмою, яка відповідає на це питання.

Вхідні дані Кожен тестовий випадок описаний за допомогою одного рядка. Рядок містить ціле число A, що представляє кут, який слід перевірити ($0 \leq A \leq 180$).

Вихідні дані

Для кожного тестового випадку виведіть рядок, що містить символ. Якщо існує, принаймні один раз в день, так що мінімальний кут між

двома стрілками годинник є рівно A градусів, то вивести велику літеру «Y». В іншому випадку вивести велику літеру "N".

Джерело ACM ICPC Regional Latino America 2012

Складність: 9% — 72/41/35/32.

Переберемо всі 720 позицій та перевіримо потрібну умову.

Програма на **Python** (160 ms, 7.7 MiB)

```
for line in open('input.txt'):
    A,f=int(line),1
    for m in range(720):
        ah=6*(m//12)
        am=6*(m%60)
        a=(ah-am)%360
        if a>180: a=360-a
        if a==A: f=0; print('Y'); break
    if f: print('N')
```

3.21.6. e7458 Гринвіцький годинник

В Гринвіцький обсерваторії жив старий астроном. Одного дня в нього зупинився годинник. І він вирішив використовуючи знання про Сонце самостійно встановити час. Він навмання завів годинник і став записувати час сходу та заходу Сонця. Астроном також знає що теоретично о 12 годині 0 хвилин Сонце повинно бути в зеніті. Але старість має свої мінуси, і астроном уже кілька днів сумнівається, чи правильно в нього відображається час на годиннику. Допоможіть астроному в його обрахунках.

Вхідні дані 4 числа — години та хвилини сходу та заходу Сонця.

Вихідні дані

2 числа — години та хвилини відхилення від правильного часу.

Input	**Output**
5 30	0 -5
18 20	

Пояснення: Годинник відстає на 5 хв.

Складність: 40% — 1196/335/443/266.

Переведемо час в мінімальні одиниці, хв. Шукане відхилення — середнє значення сходу та заходу Сонця без полудня (12:00 або 720 хв).

C++

```
#include <iostream>
using namespace std;
int main() {
        int hr,mr,hs,ms,d;
        cin >> hr >> mr >> hs >> ms;
        d=(hr+hs)*30+(mr+ms)/2-720;
        cout << d/60 << ' ' << d%60;
}
```

Аналогічний розв'язок є і в [16].

На Python враховуємо особливості дії операторів // та % **??**.

Python (28 ms, 5.4 MiB)

```
h,m = map(int,input().split())
H,M = map(int,input().split())
d=((h+H)*60+m+M)//2-720
print(d//60+(d<0)*(d%60!=0),d%60-60*(d<0)*(d%60!=0))
```

3.21.7. e1125 Кут

Інтервал кутів між годинниковою та хвилинною стрілками годинника від 0 до 180 градусів (включаючи кути 0 і 180 градусів). Наприклад, коли на годиннику 12 годин, то кут між стрілками 0 градусів, а коли 6:00, то відповідно 180 градусів. Обчисліть кут між годинниковою та хвилинною стрілками у інтервалі часу від 12:00 до 11:59.

Вхідні дані Вхідні дані містять декілька випадків. Кожен тестовий випадок у окремому рядку містить 2 цілих числа, відокремлених пропуском: перше число показує кількість годин (від 0 до 12) а друге відповідно кількість хвилин (в інтервалі [0, 59]). Вхідні дані завершуються рядком, що мітить два нулі.

Вихідні дані Для кожного тестового випадку у окремому рядку вивести мінімальний кут між стрілками в градусах у форматі, наведеному у прикладі вихідних даних.

Input	**Output**
12 0	At 12:00 the angle is 0.0 degrees.
12 30	At 12:30 the angle is 165.0 degrees.
6 0	At 6:00 the angle is 180.0 degrees.
3 0	At 3:00 the angle is 90.0 degrees.

0 0

Джерело II етап Всеукраїнської олімпіади 2010-2011 м. Бердичів
Складність: 31% — 2821/565/698/485.

Python (32 ms, 7.7 MiB)

```
h,m = map(int,input().split())
while h+m>0:
    a=abs((h%12+m/60)*30-m*6)
    if a>180: a=360-a
    print('At ',h,':','%02d'%m,' the angle is ', \
        '%.1f'%a,' degrees.',sep='')
    h,m = map(int,input().split())
```

Персоналії

Герон

Герон Александрійський [29] (*Ηρων ο Αλεξανδρευς*, Heron ho Alexandreus; також відомий як Heron of Alexandria; с. 10 AD – с. 70 AD) Невідомо точні дати народження і смерті цього давньогрецького ученого і винахідника з міста Александрії. Лише майже через 2000 років були знайдені і перекладені сучасними мовами арабські списки його праць.

Далекі нащадки дізналися, що йому належать формули визначення площі різних геометричних фігур. Найбільш відома його формула для знаходження площі трикутника (Формула Герона) (2.6.1).

Стало відомо, що Герон описав прилад діоптр, який з повною підставою можна назвати прапрадідом сучасного теодоліта. Без цього приладу не можуть зараз обійтися геодезисти, гірники, будівельники.

Він вперше дослідив п'ять типів простих машин: важіль, корбу (кривошип), клин, гвинт і блок. Герон заклав основи автоматики. Люди дивувалися дивам: двері храму самі відкривалися, коли над жертовником запалювався вогонь. Він придумав автомат для продажу «святої» води. Сконструював кулю, що обертається силою струменя пари. Винайшов ще ряд приладів і автоматів. Висунув ідею парових машин.

«Метрика» Герона і витягнуті з неї «Геометричний» і «Стереометрія» представляють собою довідники з прикладної математики. Серед відомостей в «Метриці» містяться:

Цілочисельні геронові трикутники.

Формули для площ правильних багатокутників.

Об'єми правильних багатогранників, піраміди, конуса, зрізаного конуса, тора, кульового сегмента.

Формула Герона для розрахунку площі трикутника за довжинами його сторін (відкрита Архімедом).

Правила чисельного рішення квадратних рівнянь.

Алгоритми знаходження квадратних і кубічних коренів (ітераційна формула Герона).

В основному виклад в математичних працях Герона догматичний: правила часто не виводяться, а тільки показуються на прикладах.

Книга Герона «Визначення» являє собою великий звід геометричних визначень, здебільшого збігаються з визначеннями «Начал» Евкліда.

Горнер

Вільям Джордж Горнер [33] (William George Horner, 1786, Бристоль – 22 вересня 1837) — британський математик і винахідник; він був шкільним учителем, директором і шкільним директором, знавцем класики, а також математики, який широко писав про функціональні рівняння, теорію чисел та теорію наближення, а також про оптику. Його внесок в теорії наближень відзначається в позначенні методу Горнера.

Основні праці з алгебри. У 1819 р. опублікував спосіб наближеного обчислення дійсних коренів многочлена, який називається тепер способом Руффіні-Горнера (цей спосіб був відомий китайським в XIII ст. та перським математикам сотні років тому). Робота була надрукована у Філософських роботах Королівського наукового товариства.

В XIX–початку XX століття метод Горнера займав значне місце в англійських та американських підручниках з алгебри. Де Морган показав широкі можливості метода Горнера в своїх роботах.

Іменем Горнера названа схема ділення многочлена на двочлен $x - a$.

Григорій XIII

Григорій XIII [34] (*7 січня 1502 — †10 квітня 1585) — 226-ий Папа Римський (з 13 травня 1572 року по 10 квітня 1585 року).

Уго Бонкомпаньї народився 7 січня 1502 в Болоньї. Походив з багатої дворянської сім'ї. Вивчав право у Болонському університеті, а потім, як кваліфікований юрист, доктор права, був радником багатьох єпископів в Римі і Триденті.

Григорій XIII прославив своє ім'я, ввівши в усіх католицьких країнах розроблений Луїджі Ліліо та священиком-єзуїтом і астрономом Христофором Клавієм **Григоріан-**

ський календар. Календар був введений папською буллою Inter gravissimas 24 лютого 1582 року.

Причиною реформи було те, що середня тривалість року за Юліанським календарем була занадто довгою — оскільки вона розглядалася тривалістю 365 днів, 6 годин, тоді як розрахунки показали, що фактична середня тривалість року трохи менша (365 днів, 5 год 49 хв). Як результат, дата весняного рівнодення повільно (протягом 13 століть) сповзла до 10 березня, тоді як за пасхалією (розрахунком дати Пасхи) все ще належало брати традиційну дату 21 березня.

Реформа календаря ліквідувала десятиденне відставання юліанського календаря по відношенню до сонячного року. Високосні роки, коли місяць лютий нараховує 29 днів, встановлювалися рідше (відтепер не були високосними роки, кратні 100, але не кратні 400, наприклад, 1700, 1800, 1900).

Новий календар належним чином замінив юліанський календар, який використовувався з 45 р. До н.е.

Для розуміння змін у календар, підготовки та проведення його реформи Григорій XIII у 1578 році наказав збудувати вежу для астрономічних спостережень і заснував Ватиканську обсерваторію.

Евклід

Евклід [35] (грец. *Ενκλειδηζ*, від «добра слава», Euclid, Эвклид; близько 365 — близько 270 до н.е.) — старогрецький математик і визнаний основоположник математики, автор перших теоретичних трактатів з математики, що дійшли до сучасності.

Відомо, що народився він в Афінах, жив в Александрії при Птолемеї I. Евклід заснував в Александрії свою школу.

Евклід є автором найдавніших трактатів з математики, що збереглись до сьогодення. В них підсумовано досягнення давньогрецької математики. Наукова діяльність Евкліда проходила в Александрійській бібліотеці — суспільній інституції, що являла собою бібліотечний, науковий, навчальний, інформаційно-аналітичний і культурологічний комплекс.

Основна праця Евкліда «Начала» складається із серії книжок, у яких міститься систематизований виклад геометрії, а також деяких питань теорії чисел. «Начала» відіграли виняткову важливу роль у подальшому

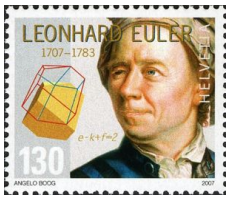
розвитку математичної науки.

Перша та деякі інші книги містять на початку списки визначень. У першій книзі подається 23 попередніх визначення об'єктів геометрії: наприклад, «точка — це те, що не має частин»; «лінія — це довжина без ширини»; «пряма лінія — це та, що однаково розташована відносно точок на ній». Уводяться визначення кута, площини, квадрата, кола, сфери, призми, піраміди, п'яти правильних многогранників тощо. Далі подано п'ять постулатів і дев'ять аксіом.

У I книзі вивчаються властивості трикутників і паралелограмів. Книга II, виходить від піфагорійців, присвячена так званій «геометричній алгебрі». У III і IV книгах висловлюється геометрія кіл, а також вписаних і описаних багатокутників. XI книга містить основи стереометрії.

Мав також роботи з астрономії, оптики, теорії музики. Сформулював закон прямолінійного поширення світла. У своїх працях розглядав утворення тіні, отримання зображення за допомогою малих отворів, явища, пов'язані з відбиттям променів від плоских і сферичних дзеркал, що дає підстави вважати Евкліда Основоположником геометричної оптики.

Ейлер



Леонард Ейлер [36], правильніше Ойлер (нім. Leonhard Euler; 15 квітня 1707, Базель, Швейцарія – 7 (18) вересня 1783, Санкт-Петербург) — швейцарський математик та фізик, який провів більшу частину свого життя в Росії та Німеччині.

Ейлер є автором 866 наукових публікацій, зокрема у галузях математичного аналізу, диференціальної геометрії, теорії чисел, теорії графів, наближених обчислень, небесної механіки, математичної фізики, оптики, балістики, кораблебудуванні, теорії музики, що мали значний вплив на розвиток науки. Саме він ввів більшість математичних понять та символів у сучасну математику, наприклад: $f(x)$, e, π, уявна одиниця i, символ суми Σ і багато інших.

Ейлер вважається найвидатнішим математиком 18-го століття, а, можливо, навіть усіх часів.

У 1736 році, Ейлер розв'язав проблему, відому як Сім мостів Кенігсберґа. Місто Кенігсберг (сьогодні Калінінград) в Пруссії розташоване на річці Преголя і включає два великі острови, які були пов'язані один з одним і з материком сімома мостами. Проблема полягає в тому, чи можна знайти шлях, який проходить кожним мостом рівно один раз і повертається до початкової точки. Відповідь є негативна: немає ци-

клу Ейлера. Це твердження вважається першою теоремою теорії графів, зокрема, в теорії планарних графів.

Ейлер також довів формулу $V-E+F=2$, що пов'язує число вершин, ребер і граней опуклого багатогранника.

Ератосфен

Ератосфен Киренский [37] (грец. *Ερατοσθενης*, Eratosthenes) (бл. 275 — 194 до н.е.), давньогрецький науковець і письменник. Один із надзвичайно різнобічних вчених античності. Ератосфен займався філологією, філософією, хронологією, математикою, астрономією, геодезією, географією, сам писав вірші і музику.

Ератосфен розрахував окружність Землі не покидаючи Єгипту.

Серед математичних творів Ератосфена виділяється твір Платоники (Platonikos), свого роду коментар до діалогу «Тімей» Платона, у якому розглядалися питання з області математики і музики. Ератосфен звертається до математичних і музичних основ платонівської філософії. Вихідним пунктом було так зване делійське питання, тобто подвоєння куба, якому автор присвятив трактат Подвоєння куба. Геометричний зміст мав твір Про середні величини (Peri mesotenon) у 2 частинах, присвячений розв'язуванню геометричних та арифметичних задач. Широко відомий трактат Решето (Koskonon). В ньому вчений виклав спрощену методику визначення простих чисел (так зване «решето Ератосфена».

фон Нейман

Джон фон Нейман [38] (в Угорщині Янош фон Нейман, Neumann János Lajos; в Німеччині Йоганн, Johann von Neumann; в США Джон, John von Neumann; 28 грудня 1903 — 8 лютого 1957) — американський математик угорського походження, що зробив значний вклад у квантову фізику, функціональний аналіз, теорію множин, інформатику, економічні науки та в інші численні розділи знання. Він став засновником теорії ігор разом із Оскаром Морґенштерном у 1944 році. Розробив архітектуру (так зва-

ну «архітектуру фон Неймана»), яка використовується в усіх сучасних комп'ютерах.

Задача про муху [32]. Два потяги зближуються з відстані 200 км зі швидкостями 50 км/год. З вітрового скла одного з локомотивів в початковий момент злітає муха і починає літати зі швидкістю 75 км/год вперед назад між поїздами, доки ті, зіткнувшись, не розчавлять її. Яку відстань встигає пролетіти муха до зіткнення?

З кожним з поїздів муха встигає зустрітись нескінченну кількість разів. Щоб знайти відстань, яку пролетіла муха, можна просумувати не нескінченний ряд відстаней (ці відстані спадають достатньо швидко, і ряд сходиться). Джон фон Нейман, коли ему задали цю задачу, задумався лише на мить і: "Ясна річ, 150 км!". Приятель спитав його: "Як Вам вдалось так швидко отримати відповідь?— "Я просумував ряд — пошуткував математик.

Паскаль

Блез Паскаль [39] (Blaise Pascal; 19 червня 1623, Клермон-Ферран – 19 серпня 1662, Париж) — французький філософ, письменник, фізик, математик.

Один із засновників математичного аналізу, теорії ймовірностей та проективної геометрії, творець перших зразків лічильної техніки, автор основного закону гідростатики. Відомий також відкриттям формули біноміальних коефіцієнтів, винаходом гідравлічного преса й шприца та іншими відкриттями. Автор знаменитих «Думок» та «Листів до провінціала», які стали класикою французької літератури.

На честь Паскаля названа одиниця вимірювання тиску (Паскаль), а також популярна мова програмування Pascal.

У 1642 році (у 19 років) Паскаль почав створення своєї Лічильної машини «паскаліни», в цьому, за його власним визнанням, йому допомогли знання, отримані в ранні роки. Машина Паскаля виглядала як скринька, наповнена численними, пов'язаними одна з одною зубчастою передачею, шестернями. Числа, які треба було додати або відняти, вводилися відповідним поворотом коліщат, принцип роботи ґрунтувався на рахунку обертів. Винайдений Паскалем принцип пов'язаних коліщат майже на три століття став основою створення більшості арифмометрів.

Паскаль підтвердив припущення Еванджеліста Торрічеллі про існування атмосферного тиску. Розвиваючи результати досліджень Стевіна

і Галілея в сфері гідростатики у своєму «Трактаті про рівновагу рідин» (1653, опублікований в 1663), Паскаль підійшов до встановлення закону розподілу тиску в рідинах. У другому розділі трактату він формує ідею гідравлічного преса/ Почавши з простого повторення досліду Торрічеллі, Паскаль спростував одну з основних аксіом старої фізики і встановив основний закон гідростатики.

Паскалем, Ферма і Робервалем в листуванні Паскаля з Ферма, закладаються основи теорії імовірності. Вчені, під час розв'язання задачі про розподіл ставок між гравцями при перерваній серії партій, використовували кожен свій аналітичний метод підрахунку ймовірностей і прийшли до однакового результату. Інформація про пошуки Паскаля і Ферма підштовхнула Гюйгенса до заняття проблемами імовірності, який сформулював у своєму творі «Про розрахунки в азартних іграх» (1657) визначення математичного очікування.

Паскаль створює «Трактат про арифметичний трикутник» (виданий в 1665 році), де досліджує властивості «трикутника Паскаля» та його застосування в підрахунку числа сполук, не вдаючись до алгебраїчних формул. Одним з додатків до трактату була робота «Про підсумовування числових степенів», де Паскаль пропонує метод підрахунку степенів чисел натурального ряду.

Фундаментальне дослідження циклоїди (як розповідали друзі, він зайнявся цією проблемою, щоб відволіктися від зубного болю). За одну ніч Паскаль розв'язав задачу Мерсенна про циклоїди і зробив низку відкриттів у її вивченні.

Піфагор

Піфагор [40] (*Πυθαγοραζ*, 570 до н. е., Сідон — 497 до н. е., Метапонт) — давньогрецький філософ, релігійний та політичний діяч, засновник піфагореїзму, який став легендою і джерелом дискусій уже в стародавні часи. У 306 р. до н. е. йому, як найрозумнішому з греків, поставили пам'ятник у Римському Форумі.

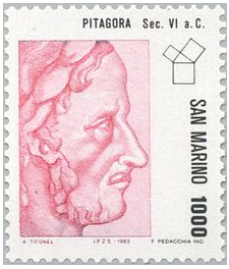
Введення терміну «філософ» приписують Піфагору, який назвав себе не мудрецем, а «тим, хто любить мудрість»

Піфагор займає почесне місце в історії математики. Він відкрив нову епоху в еволюції наукової думки. Піфагорійці перетворили давно відомі практичні правила в наукові положення, обґрунтовані точними доведе-

ннями. Піфагор увів загальновизнаний тепер дедуктивний метод, суть якого полягає в тому, що, крім невеликої кількості прийнятих без доведень первісних положень, які називаються аксіомами, всі інші твердження математики виводяться логічними міркуваннями.

Основним змістом піфагорійської математики є вчення про число. Як і вавилонські маги, піфагорійці вважали надзвичайно важливими різні властивості чисел і відношення між ними. І коли відсіяти полову — числову містику, виявиться, що вони ввели багато фундаментальних теоретико-числових понять, виявили і дослідили глибокі властивості чисел і поставили такі питання, які й сьогодні залишаються предметом досліджень багатьох учених і все ще чекають свого розв'язання.

Найважливішою властивістю чисел піфагорійці вважали парність і непарність і першими ввели поняття парного і непарного числа, простого і складеного, розробили теорію подільності на два, дали кілька класифікацій натуральних чисел. Піфагорійці вважали унікальними такі числа, в яких сума власних дільників, тобто дільників, менших від самого числа, дорівнює самому числу (досконалі числа).



Почувши ім'я Піфагора, ми відразу пригадуємо знамениту теорему: «Сума квадратів катетів дорівнює квадрату гіпотенузи». А втім, якщо Піфагор справді був у Вавилоні, він міг довідатися про те, що шумеро-вавилонські математики знали і використовували під час розв'язування задач теорему, названу пізніше його ім'ям, десь за 1500 років до народження Піфагора. Присвоєння цій знаменитій теоремі імені Піфагора свідчить про те, якого значення надавали в той час доведенню математичних тверджень. Піфагорові приписують й інші теореми: про суму внутрішніх кутів трикутника; про те, що площину навколо точки можна заповнити лише трьома видами правильних многокутників: рівносторонніми трикутниками, квадратами і правильними шестикутниками. Можливо, він знав теорему про те, що площі подібних фігур відносяться, як квадрати відповідних сторін, і був обізнаний з трьома правильними многогранниками: тетраедром, кубом і додекаедром. Сторонами додекаедра є правильні п'ятикутники. Якщо в правильному п'ятикутнику провести всі діагоналі, дістанемо піфагорійську зірку, або пентаграму, — улюблену фігуру піфагорійців.

У школі Піфагора було зроблено відкриття, яке започаткувало нову епоху в історії математики і завдало нищівного удару по піфагорійських поглядах на число. Шукаючи спільну міру між стороною і діа-

гоналлю квадрата, піфагорійці відкрили, що ці відрізки спільної міри не мають, тобто не існує числа, яким би можна було виразити відношення цих відрізків. Доведення піфагорійцями несумірності сторони і діагоналі квадрата — перше відоме в історії математики доведення «від супротивного» (міркування, при якому виходять з істинності твердження, протилежного доводжуваному, і приходять до суперечності). Метод доведення «від супротивного» розробив визначний давньогрецький математик Евдокс Кнідський.

Фібоначчі

Леонардо Пізанський [41] (Leonardo Pisano, близько 1170 — близько 1250), відоміший як Фібоначчі (Fibonacci) — італійський математик 13 століття, автор математичних трактатів, завдяки яким Європа довідалася про вигадану індійцями позиційну систему числення, відому зараз як арабські цифри. Леонардо розглянув також ідею так званих чисел Фібоначчі і вважається одним з найвидатніших західних математиків Середньовіччя.

Значну частину засвоєних ним знань він виклав у своїй видатній «Книзі абака» (Liber abaci, 1202; до наших днів зберігся тільки доповнений рукопис 1228 року). Ця книга містить майже всі арифметичні й алгебраїчні відомості того часу, викладені з винятковою повнотою і глибиною. Вона відіграла значну роль у розвитку математики в Західній Європі протягом кількох наступних століть. Саме за цією книгою європейці знайомилися з арабськими цифрами. Перші п'ять розділів книги присвячено арифметиці цілих чисел на основі десяткової системи числення.

"Практика геометрії"(Practica geometriae, 1220) містить різноманітні теореми, пов'язані з вимірювальним методом.

У трактаті "Квітка"(Flos, 1225) Фібоначчі досліджував задачу, яка в сучасних позначеннях зводиться до знаходження коренів кубічного рівняння.

"Книга квадратів"(Liber quadratorum, 1225), містить ряд задач на знаходження розв'язку невизначених квадратних рівнянь.

Задачі Фібоначчі

Задачі про гирі

Завдання про вибір найкращої системи гир для зважування на важільних терезах вперше була сформульована саме Фібоначчі. Леонардо

Пізанський пропонує два варіанти завдання:

Простий варіант: потрібно знайти п'ять гир, за допомогою яких можна знайти вагу меншу ніж 30, при цьому гирі можна класти лише на одну чашу терезів (Відповідь: 1, 2, 4, 8, 16). Розв'язок будується в двійковій системі числення.

Складний варіант: потрібно знайти найменше число гир, за допомогою якого можна зважити вагу меншу від заданої (Відповідь: 1, 3, 9, 27, 81,...). Розв'язок будується в системі числення з основою три.

Задача про птахів і вежі

Дві пташки одночасно злітають з вершин двох веж, що знаходяться на відстані 50 метрів. Висота однієї вежі становить 30 метрів, а другої 40 метрів. При польоті з однаковою швидкістю пташки долітають одночасно до фонтану, що розташований на лінії, проведеній через дві вежі (на рівні поверхні ґрунту). На якій відстані від основи веж знаходиться фонтан?

Задача про купця з Пізи

Пізанський купець під час торговельної подорожі до Венеції подвоїв там свій стартовий капітал, а за тим витратив 12 динарів. Потім подався до Флоренції, де знову подвоїв число своїх динарів і витратив 12. Після повернення до Пізи черговий раз подвоїв свій статок, витратив 12 динарів і ... залишився без копійки. Скільки динарів він мав на початку?

Задача про трьох чоловіків і знайдений гаманець

Три чоловіки знайшли гаманець із 23 динарами. Перший каже другому: "Якщо я додам ці гроші до своїх, то буду мати суму, що буде у два рази більшою від тієї, що є у тебе". Другий аналогічно звернувся до третього: "Якщо я зараз візьму ці гроші собі, буду мати суму у три рази більшу від твоєї". На кінець третій каже до першого: "Якщо я додам ці гроші до своїх, то буду мати суму у чотири рази більшу, ніж у тебе". Скільки динарів мав кожен з них?

Загадка пана з Палермо

Три придворних слуги мали свою частку в певній сумі грошей у касі: доля першого становила половину, другого — третину, а третього — шосту частину. Кожен з учасників взяв зі спільної каси гроші не зовсім чесно так, що каса залишилась порожньою. Далі перший з них повернув половину того, що взяв, другий — третину, а третій — шосту частину. Отриману суму було розділено на три однакові частини і роздано кожному з трьох слуг. Виявилось, що кожен з них мав точно стільки, скільки йому належало. Скільки коштів було у касі спочатку, яку суму взяв кожен з учасників?

Задача про спадщину

Чоловік, що помирав покликав своїх синів і сказав найстаршому: «Візьми одного динара з моїх статків і сьому частину від того, що залишиться». Другому сину каже: «Візьми собі два динари і сьому частину того, що залишиться». До третього: «Візьми три динари і сьому частину того, що залишиться» і так далі — кожному наступному синові записував на один динар більше від попереднього і сьому частину залишку. Після поділу статків виявилось, що всі сини отримали порівну. Скільки було синів і яким був спадок?

Задачі з теорії чисел

Знайти число, яке ділиться на 7 і дає в залишку одиницю при діленні на 2, 3, 4, 5 і 6;

Знайти число, добуток якого на сім дає залишки 1, 2, 3, 4, 5 при діленні на 2, 3, 4, 5, 6, відповідно;

Знайти квадратне число, яке при збільшенні або зменшенні на 5 давало б квадратне число.

Предметний покажчик

Бібліографія

[1] Офіційний сайт ICPC. — Режим доступу: https://icpc.baylor.edu/.

[2] ICPC, Вікіпедія. — Режим доступу: https://en.wikipedia.org/wiki/International_Collegiate_Programming_Contest.

[3] BAPC, Вікіпедія. — Режим доступу: https://en.wikipedia.org/wiki/BAPC.

[4] Офіційний сайт BAPC. — Режим доступу: https://2019.bapc.eu.

[5] Офіційний сайт IOI. — Режим доступу: https://ioinformatics.org.

[6] Всеукраїнські олімпіади з інформатики. — Режим доступу: https://oi.in.ua.

[7] NetOI центр підтримки та проведення Всеукраїнських олімпіад школярів з інформатики в мережі Інтернет. — Режим доступу: https://new.netoi.org.ua.

[8] acmp.ru. — Режим доступу: https://acmp.ru.

[9] e-olymp.com. — Режим доступу: https://www.e-olymp.com.

[10] codeforces. — Режим доступу: https://codeforces.com.

[11] Timus Online Judge. — Режим доступу: https://acm.timus.ru.

[12] Алготестер. — Режим доступу: https://algotester.com/uk.

[13] ACM Контестер. — Режим доступу: http://acm.lviv.ua/fusion/news.php.

[14] ideone. — Режим доступу: https://ideone.com/.

[15] OnlineGDB online compiler and debugger for c/c++. — Режим доступу: https://www.onlinegdb.com.

[16] Матвійчук С.В., Жуковський С.С. Практикум програмування Python / C++ на e-olymp.com. — Житомир : Вид. О.О. Євгенок, 2019. — 232 с.

[17] Лааксонен Антти. Олимпиадное программирование. / пер. с англ. А. А. Слинкин. — М. : ДМК Пресс, 2018. — 300 с.

[18] Меньшиков Ф. Олимпиадные задачи по программированию. — СПб. : Питер, 2006. — 315 с. — Режим доступу: https://acmp.ru/article.asp?id_text=42627.

[19] Долинский М.С. Решение сложных и олимпиадных задач по программированию: Учебное пособие. — СПб. : Питер, 2006. — 366 с.

[20] Арлазаров В.Л., Мамай И.Б. Спортивное программирование: метод. пособие по подготовке к олимпиадам школьников. 7–11-й классы. — М. : Изд. Дом МИСиС, 2017. — 28 с.

[21] Симоненко Е.А. Спортивное программирование. Сборник рецептов для программирующих на C++. — Краснодар, 2012. — 29 с.

[22] Андреева Е.В., Гурвиц В.М., Матюхин В.А. Московские олимпиады по информатике. — М. : МЦНМО, 2006. — 256 с.

[23] Ускова О.Ф., Горбенко О.Д., Шашкин А.И. Олимпиадные задачи по программированию и лучшие решения. Часть 1. — Воронеж : ООО ПФ «Джуди», 2001. — 76 с.

[24] Шень А. Программирование: теоремы и задачи. 6-е изд., дополненное. — М. : МЦНМО, 2017. — 320 с.

[25] C++ для приматов. Начальный курс программирования, ОНУ ім. І.І.Мечнікова. — Режим доступу: https://cpp.mazurok.com.

[26] Решение задач с e-olymp. — Режим доступу: https://devexe.top/eolimp.

[27] Решение задач с сайта e-olimp.com. — Режим доступу: https://sites.google.com/site/reshenieolimp/.

[28] Теорія чисел, Вікіпедія. — Режим доступу: https://en.wikipedia.org/wiki/Number_theory.

[29] Герон Александрійський, Вікіпедія. — Режим доступу: https://en.wikipedia.org/wiki/Hero_of_Alexandria.

[30] A+B, Вікіпедія. — Режим доступу: https://ru.wikipedia.org/wiki/A+B.

[31] Приклади коду до задачі різними мовами програмування, e-olymp. — Режим доступу: https://www.e-olymp.com/uk/problems/1/discussion.

[32] Федин С.Н. Математики тоже шутят. — ЛИБРОКОМ, 2009. — 111 с.

[33] Горнер, Вікіпедія. — Режим доступу: https://en.wikipedia.org/wiki/William_George_Horner.

[34] Григорій XIII, Вікіпедія. — Режим доступу: https://en.wikipedia.org/wiki/Pope_Gregory_XIII.

[35] Евклід, Вікіпедія. — Режим доступу: https://en.wikipedia.org/wiki/Euclid.

[36] Леонард Ейлер, Вікіпедія. — Режим доступу: https://en.wikipedia.org/wiki/Leonhard_Euler.

[37] Ератосфен, Вікіпедія. — Режим доступу: https://en.wikipedia.org/wiki/Eratosthenes.

[38] фон Нейман, Вікіпедія. — Режим доступу: https://en.wikipedia.org/wiki/John_von_Neumann.

[39] Блез Паскаль, Вікіпедія. — Режим доступу: https://en.wikipedia.org/wiki/Blaise_Pascal.

[40] Піфагор, Вікіпедія. — Режим доступу: https://en.wikipedia.org/wiki/Pythagoras.

[41] Фібоначчі, Вікіпедія. — Режим доступу: https://en.wikipedia.org/wiki/Fibonacci.

Навчально-методичне видання

Жмурко Олександр Іванович,
Охріменко Тетяна Олександрівна

Олімпіади з програмування
Прості задачі

Видається в авторській редакції
Підписано до друку . .2020 р. Формат 60×84/16
Папір офсетний. Ум.друк.арк.17,12
Тираж 300 прим. Замовлення №

Видавничо-поліграфічний центр «Візаві»
20300, м.Умань, вул Тищенка, 18/19, вул. Садова, 2
Свідоцтво суб'єкта видавничої справи
ДК № 2521 від 06.08.2006.

тел.(04744)4-64-88, 3-51-33, (067)104-64-88
vizavi-print.jimdo.com
e-mail: vizavi@gmail.com
vizavisadova@gmail.com