туації нескладно вийти, замінивши натомість IA на AB, а першим правилом у програмі — B на десять I.

Підзадача 5. Спочатку встановимо індикатор початку рядка: замінимо всі a на 0a, b на 0b, c на 0c, а всі a0 на a, b0 на b, а c0 — на c. Після таких операцій у нас залишиться єдиний нуль на першій позиції. Тепер будемо робити великими літери, що опинилися на початку рядка, тобто замінювати 0a на 0A, 0b на 0B, 0c на 0C. Водночас будемо переставляти всі 9 пар з великих та малих літер, коли велика стоїть перед малою. Так ми спочатку пересунемо першу літеру в кінець, потім друга літера стане на передостаннє місце і т. д. Насамкінець лишається замінити 0 на порожній рядок. Єдина проблема тут така, що при початковій зміні, наприклад, символу a на 0a послідовність операцій буде зациклюватися. Тому мінятимемо a на 0x, b на 0y, c на 0z, після чого ототожнюватимемо x із a, y із b, а z — із c.

Підзадачі 6 і 7. Ідея базується на розкладі обох доданків у відповідну кількість літер I, після чого літери (яких тепер якраз сумарна кількість) збираються назад у число. Основна проблема полягає в тому, що після того, як ми зберемо остаточне число, рядковий автомат знов захоче розкладати числа в літери I . Щоб цього не сталося, ми маємо використати єдину відмінність між вхідними та потенційними вихідними даними: наявність плюса. Локальність цього плюса можна побороти таким чином: замінюватимемо, скажімо, не 4 на AIIII, а 4+ на pAIIII+ та аналогічно +4 на +AIIIp. Тоді за допомогою символу p ми зможемо пропагувати зміни (а точніше, уникати зайвих змін) також і значно лівіше та правіше від плюса. Після цього плюс та символ p прибираються та йде зворотний процес. Проблеми з розміром рядка заміни вирішуємо так само, як і в четвертому пункті. Цілком імовірно, що кількість виконуваних операцій при цьому вийде за допустиму межу. Тоді можна оптимізувати програму: додатково до заміни IA на AB (див. четверту підзадачу) введемо заміни IIA на ABB, IIIA на ABBB і т. д. Щоб перша частина алгоритму (зведення до одиниць) не конфліктувала з другою (зведення до десяткового запису), можемо в першій частині за одиниці брати літери I, далі міняти їх на J, а в другій частині одиницями вважати вже літери J, а I ніяк не зачіпати. Щоб оптимізувати кількість операцій, потрібно ввести не лише заміну I на J, але й, наприклад, II на JJ, III на JJJ і т. д. Насамкінець зауважимо, що в третьому пункті ми працювали лише з числами в межах до 1000, а в даній підзадачі сума може вийти більшою. Отже, потрібно доповнити відповідні правила для коректної роботи з числами до 2000.

* * *

ОЛІМПІАДА З ІНФОРМАТИКИ У МІСТІ КИЄВІ У 2014–2015 НАВЧАЛЬНОМУ РОЦІ

Мисак Данило Петрович,
керівник гуртка СШ №52 м. Києва.
Рибак Олександр Владиславович,
аспірант Інституту математики НАН України.
Рудик Олександр Борисович,
доцент Київського університету імені Бориса Грінченка.

Продовження, початок у №4 за 2015 рік

IV. УМОВИ ЗАВДАНЬ III ЕТАПУ

1. Дивовижне число (автор — Данило Мисак)

Максимальна оцінка: 200 балів
Обмеження на час: 2 сек.
Обмеження на пам'ять: 256 МБ
Вхідний файл: awesome.in
Вихідний файл: awesome.out
Програма: awesome.*

Якось Орися з Марисею натрапили на деяке натуральне число n. Орися порахувала суму його цифр, а Марися — добуток. На подив дівчатам пораховані сума та добуток виявилися рівними одному й тому самому числу d.

Завдання. Знаючи d, відновіть найбільше можливе значення початкового числа n.

Вхідні дані

У вхідному файлі вказано натуральне число d.

- У 50% тестів $d \le 20$.
- У 50% тестів $20 < d \le 1\,000\,000$.

Усі тести мають однакову вартість.

Вихідні дані

У вихідний файл виведіть найбільше можливе значення n або число 0, якщо такого значення не існує.

Приклади

awesome.in	awesome.out
6	321
7	7
11	0

Пояснення до першого прикладу

І сума, і добуток цифр числа 321 дорівнюють 6: 3+2+1=3·2·1=6. Таку саму властивість мають і числа 312, 231, 213, 132, 123, а також одноцифрове число 6, але 321 з них найбільше, тому саме його і треба вивести.

2. Черевички на підборах (автор — Данило Мисак)

Максимальна оцінка: 200 балів
Обмеження на час: 1,5 сек.
Обмеження на пам'ять: 256 МБ
Вхідний файл: heels.in
Вихідний файл: heels.out
Програма: heels.*

Орися та Марися збирають черевички на підборах: Орися ліві, а Марися — праві. Між собою черевички відрізняються лише висотою підбора. В обох дівчат назбиралося рівно по n черевиків. Деякі з черевиків, навіть якщо вони на одну й ту саму ногу, можуть мати однакову висоту підбора.

Завдання. Знаючи висоту підборів усіх лівих та всіх правих черевиків, з'ясуйте, скільки повноцінних пар взуття з них можна скласти. З лівого та правого черевика можна скласти пару, якщо висота підборів у них однакова.

Вхідні дані

У першому рядку вхідного файлу вказано натуральне число n. У другому рядку записано n натуральних чисел, що задають висоти підборів черевиків на ліву ногу. У третьому рядку записано n натуральних чисел, що задають висоти підборів черевиків на праву ногу. Висота жодного підбора не перевищує 10^9.

- У 25% тестів $n \le 1000$ і в жодної дівчини немає двох черевиків з однаковою висотою підбора.
- У 25% тестів $n \le 1000$ і принаймні в однієї з дівчат є хоча б два черевики з однаковою висотою підбора.
- У 25% тестів $1000 < n \le 200\,000$ і висоти підборів не перевищують 1000.
- У 25% тестів $1000 < n \le 200\,000$ і висота підбора принаймні на одному черевикові більша за 1000.

Усі тести мають однакову вартість.

Вихідні дані

Єдиний рядок вихідного файлу повинен містити кількість пар, які можна скласти з Орисиних лівих та Марисиних правих черевичків.

Приклад

heels.in	heels.out
7 6 533 1 58 5 2 83555	4

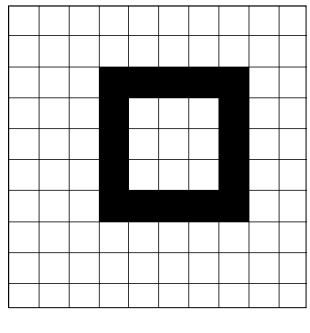
Пояснення до прикладу

З черевиків дівчат можна скласти чотири пари: пару з висотою підборів 3, пару з висотою підборів 8 і дві пари з висотою підборів 5.

3. Два квадрати (автор — Данило Мисак)

Максимальна оцінка: 200 балів
Обмеження на час: 2,5 сек.
Обмеження на пам'ять: 256 МБ
Вхідний файл: squares.in
Вихідний файл: squares.out
Програма: squares.*

В Орисі й Марисі є великий клітчастий аркуш паперу $n \times n$. Орися дуже товстим чорним маркером накреслила на ньому межі квадрата таким чином, що вони проходять не по межах клітинок, а по самих клітинках. З того ж боку аркуша свій квадрат тим же чорним маркером накреслила й Марися, причому її квадрат міг дотикатися, перетинатися або навіть збігатися з Орисиним. Розміри квадратів дівчат могли бути однаковими, але могли й відрізнятися. Відомо, однак, що розміри зовнішнього контуру обох квадратів не менші за 3×3 (тобто квадрат не вироджується в чотири чорних клітинки).



Завдання. За інформацією, як виглядає аркуш, визначте розміри й розташування на ньому обох квадратів.

Вхідні дані

У першому рядку вхідного файлу вказано натуральне число n — лінійний розмір аркуша. Далі йде n рядків, у кожному з яких *через пробіл* задано n символів, кожен з яких є або крапкою, що позначає незафарбовану клітинку, або символом #, що позначає зафарбовану клітинку. Після останнього символу кожного рядка (крапки чи #) пробілу *немає*.

- У 25% тестів $3 \le n \le 10$.
- У 75% тестів $10 < n \le 1000$.

Усі тести мають однакову вартість.

Остаточну оцінку за цю задачу буде отримано таким чином: сумарний бал за пройдені тести округлять униз до найближчого числа, що ділиться на 40. Наприклад, якщо ваша програма пройде 100% тестів, вона набере 200 балів; якщо ваша програма пройде 95% тестів, вона набере 160 балів; якщо програма пройде 45% тестів, вона набере 80 балів; якщо програма пройде 10% тестів, вона набере 0 балів.

Вихідні дані

Нумерація клітинок починається з лівого верхнього кута. Ліва верхня клітинка аркуша має координати (1,1), клітинка праворуч від неї — (2, 1), клітинка знизу — (1, 2) і т. д.

У вихідному файлі мають міститися два рядки по чотири числа, записаних через пробіл: кожен рядок задає відповідний квадрат. Перші два числа рядка задають координати лівої верхньої клітинки квадрата, а наступні два числа — координати правої нижньої клітинки. Самі рядки повинні бути розташовані в лексикографічному порядку, тобто першим іде рядок з меншим першим числом; якщо перші числа однакові, то з меншим другим числом; якщо другі числа теж однакові, то з меншим третім числом; якщо й треті числа однакові, то з меншим четвертим числом; якщо всі числа однакові, порядок ролі не грає.

У випадку, коли можливих розташувань квадратів декілька, виведіть інформацію про довільне одне з них. Вхідні дані гарантують наявність принаймні одного розташування, що задовольняє умову задачі.

Приклад

squares.in	squares.out
10 . . . . # # # # # # . . . . # . . . . # . . . . # . . . . # . . # # # # # . . # . . # . # . # . . # . . # . # # # # # # . . # . . . # . . . . . # # # # #	2 6 6 10 4 3 9 8

4. Схил (автор — Олександр Рибак)

Максимальна оцінка: 200 балів
Обмеження на час: 0,5 сек.
Обмеження на пам'ять: 32 МБ
Вхідний файл: slope.in
Вихідний файл: slope.out
Програма: slope.*

У стіну забито багато цвяхів. Орися і Марися намагаються прив'язати натягнуту нитку до двох цвяхів так, щоб отримати якомога менший, *але відмінний від нуля* кут нахилу її до горизонталі.

Завдання. Серед даних точок координатної площини знайти дві точки, які є кінцями відрізка з найменшим *додатним* кутом нахилу відносно горизонталі.

Вхідні дані

Перший рядок містить натуральне число n ($2 \le n \le 100\,000$) — кількість точок, де розташовано цвяхи. Далі ідуть n рядків, кожен з яких містить два цілих числа x_j, y_j ($0 \le x_j, y_j \le 30\,000$), що є прямокутними координатами *різних* точок. Число x_j задає розташування по горизонталі, а y_j — по вертикалі.

Вихідні дані

В єдиному рядку через пробіли вивести числа x_j, y_j, x_k, y_k. Тут (x_j, y_j) та (x_k, y_k) — координати двох точок з даних, які задають *найпологіший схил*. Інакше кажучи, при яких відношення $|x_j - x_k|/|y_j - y_k|$ *визначене* і є *найбільшим* серед відношень такого вигляду.

Точки вказати в такому порядку, щоб справджувалася нерівність $y_j < y_k$ і:

- якщо можливих розв'язків декілька, обрати той, де y_j є найменшим;
- якщо й за цієї умови є кілька варіантів, обрати розв'язок з найменшим x_j;
- якщо й за цих умов (попарного збігу перших двох чисел) є кілька варіантів, обрати розв'язок з найменшим y_k;
- якщо й за цих умов є кілька варіантів, обрати розв'язок з найменшим x_k.

Якщо задача не має розв'язку, вивести 0.

Приклади

slope.in	slope.out
4 7 4 9 2 5 1 1 2	5 1 1 2
2 123 45 256 45	0

5. Каса (автор — Олександр Рудик)

Максимальна оцінка: 200 балів
Обмеження на час: 10 сек.
Обмеження на пам'ять: 32 МБ
Вхідний файл: cash.in
Вихідний файл: cash.out
Програма: cash.*

Марися й Орися пішли до кінотеатру. Біля каси кінотеатру $m+n$ дітей (всього разом з Марисею та Орисею) вишикувалося у чергу за квитками, причому m з них мають лише по банкноті вартістю 20 гривень, а решта n мають лише по банкноті вартістю10 гривень. Вартість квитка на дитячий сеанс складає 10 гривень. На початку роботи в касі є p банкнот вартістю по 10 гривень.

Завдання. Створити програму cash.*, яка визначить кількість способів надходження банкнот у касу таким чином, щоб жодна дитина не чекала на здачу, або лишок від ділення цієї кількості на певне натуральне число r.

Вхідні дані

Вхідний файл містить *натуральні* числа m, n і *невід'ємні цілі* числа p, r:

$m \le n+p$; $m+n \le 32767$; $p \le 32767$; $r < 10^9$.

Вихідні дані

Вихідний файл має містити лише одне число:

- при $r=0$ — кількість *різних* послідовностей банкнот по 10 і 20 гривень, при яких жодній дитині не доведеться чекати здачі (вхідні дані гарантують, що у цьому випадку вказана кількість менша від 10^{18});
- при $r>0$ — лишок від ділення на r такої кількості.

Приклади

cash.in	cash.out
2 3 0 0	5
2 3 0 3	2
2 3 2 0	10
2 3 2 7	3

Пояснення до перших двох прикладів

(10, 10, 10, 20, 20), (10, 10, 20, 10, 20), (10, 10, 20, 20, 10), (10, 20, 10, 20, 10), (10,20,10, 10, 20) — вичерпний перелік прийнятних послідовностей надходження банкнот у касу, у якій на початку роботи немає нічого, а біля каси зібралося 5 дітей з трьома банкнотами по 10 гривень і двома банкнотами по 20 гривень.

6. Гра (автор — Олександр Рудик)

Максимальна оцінка: 200 балів
Обмеження на час: 0,1 сек.
Обмеження на пам'ять: 32 МБ
Вхідний файл: game.in
Вихідний файл: game.out
Програма: game.*

Марися й Орися грають у таку гру. Скінченну кількість фішок розташовано в ряд і занумеровано послідовними натуральними числами, починаючи з 1. Два гравці по черзі забирають довільну натуральну кількість фішок від 1 до певного натурального g, *розташованих поруч*. Інакше кажучи, номери забраних за один хід фішок — послідовні натуральні числа. Переможцем вважають того, хто зробить останній хід.

Завдання

Створіть програму game.*, яка для довільної позиції гри визначає всі виграшні ходи — такі ходи, що гарантують виграш (за умови правильного продовження гри зі свого боку) незалежно від ходів суперника.

Вхідні дані

Вхідний файл містить 2 рядки. У першому рядку — *натуральне* число g у межах від 2 до 9. У другому рядку записано у порядку зростання *натуральні* номери наявних фішок, які менші за 256 (щонайменше 7 чисел).

Вихідні дані

Вихідний файл має містити g рядків, кожний з яких закінчується ознакою кінця рядка. J-ий рядок вихідного файлу має містити у довільному порядку номери фішок, забравши які разом з *наступними сусідніми* $(j-1)$ фішкою гравець робить виграшний хід з позиції, заданої вхідними даними. Якщо таких ходів немає, то відповідний рядок вихідного файлу порожній. Зайвими пропусками при перевірці буде знехтувано.

Приклади

game.in	game.out
2 1 2 3 4 6 7 8	1 2 3 4 6 7
6 2 3 4 5 6 7 8 9	5 4 3

(Далі буде)