

## Задача А. Киноманы

Имя входного файла: `cinemaholics.in`  
Имя выходного файла: `cinemaholics.out`  
Ограничение по времени: 2 секунды  
Ограничение по памяти: 256 мегабайт

Макс и Мел — заядлые киноманы. Они оба посмотрели все топ-100 фильмов по версиям большинства популярных сайтов, могут перечислить фильмографию любого известного актера, и редкий их диалог не заканчивается спором, стрелял ли Хан Соло первым.

Одним прекрасным октябрьским утром Макс и Мел решили выяснить, кто же из них больше времени уделяет своему хобби. Макс подсчитал, что за последний год он посмотрел  $t_1$  трейлеров,  $e_1$  сериалов и  $f_1$  фильмов. Мел же посмотрел  $t_2$  трейлеров,  $e_2$  сериалов,  $f_2$  фильмов. Каждый трейлер идет три минуты, сериал 20 минут, а фильм два часа. Помогите им разрешить спор и определить, кто же за последний год посмотрел больше минут видео.

### Формат входных данных

В первой строке входного файла даны 3 целых числа  $t_1, e_1, f_1$  — количество трейлеров, сериалов и фильмов, которые посмотрел Макс ( $0 \leq t_1, e_1, f_1 \leq 1000$ ).

В первой строке входного файла даны 3 целых числа  $t_2, e_2, f_2$  — количество трейлеров, сериалов и фильмов, которые посмотрел Мел ( $0 \leq t_2, e_2, f_2 \leq 1000$ ).

### Формат выходных данных

Если Макс суммарно провел больше времени за просмотром видео, чем Мел, выведите **Max**.

Если Мел суммарно провел больше времени за просмотром видео, чем Макс, выведите **Mel**.

Если же они оба суммарно просмотрели одинаковое количество минут видео, выведите **Draw**.

### Примеры

| <code>cinemaholics.in</code> | <code>cinemaholics.out</code> |
|------------------------------|-------------------------------|
| 15 1 3<br>1 3 3              | Max                           |
| 20 3 0<br>0 0 1              | Draw                          |
| 1 2 3<br>3 2 1               | Max                           |

## Задача В. Макс и расстояния

Имя входного файла: `distances.in`  
Имя выходного файла: `distances.out`  
Ограничение по времени: 2 секунды  
Ограничение по памяти: 256 мегабайт

Макс нашел массив  $x$  из  $n$  целочисленных точек на прямой, упорядоченный по неубыванию. Также у Макса есть две перестановки чисел  $a$  и  $b$  от 1 до  $n$ , которые ему подарили на день рождения.

Макс решил поиграть с ними: он сгенерировал матрицу расстояний  $d$ , каждый элемент которой равен  $d_{i,j} = |x_{a_i} - x_{b_j}|$ , то есть  $d_{i,j}$  элемент равен расстоянию между  $a_i$  и  $b_j$  точкой.

Макс еще не успел наиграться с массивами, как после очередной уборки Кэти они куда-то потерялись. Все до одного: и  $x$ , и  $a$ , и  $b$ . Макс решил пойти на экстренные меры: у него сохранилась матрица расстояний  $d$ , и он хочет восстановить какой-то неубывающий массив  $x$ , а также две перестановки  $a$  и  $b$ , с помощью которых можно сгенерировать матрицу расстояний, равную  $d$ . Однако, могло случиться так, что Макс ошибся при подсчете  $d$ , и восстановить  $x$ ,  $a$  и  $b$  не получится.

Помогите Максусу восстановить их.

### Формат входных данных

В первой строке находится натуральное число  $n$  ( $1 \leq n \leq 1000$ ). В следующих  $n$  строках находится матрица  $d$  ( $0 \leq d_{i,j} \leq 10^9$ ). Все числа в матрице — целые числа.

### Формат выходных данных

В первой строке выведите «YES», если существуют такие  $x$ ,  $a$  и  $b$ , с помощью которых можно сгенерировать матрицу, равную  $d$ ,

во второй строке — неубывающий массив целых чисел  $x$  ( $|x_i| \leq 10^9$ ;  $x_i \leq x_{i+1}$ ),

в третьей строке — перестановку чисел  $a$  от 1 до  $n$ ,

в четвертой строке — перестановку чисел  $b$  от 1 до  $n$ .

Если таких  $x$ ,  $a$  и  $b$  не существует — выведите «NO».

### Примеры

| distances.in                                  | distances.out                        |
|-----------------------------------------------|--------------------------------------|
| 4<br>3 2 1 0<br>2 1 0 1<br>1 0 1 2<br>0 1 2 3 | YES<br>0 1 2 3<br>1 2 3 4<br>4 3 2 1 |
| 3<br>1 0 2<br>1 1 1<br>1 1 0                  | NO                                   |
| 4<br>7 0 2 1<br>9 2 0 3<br>0 7 9 6<br>6 1 3 0 | YES<br>0 2 3 9<br>2 1 4 3<br>4 2 1 3 |

### Замечание

Обратите внимание, что входные данные могут иметь достаточно большой объем, поэтому рекомендуется использовать быстрые потоки ввода-вывода вашего языка.

## Задача С. Многочлены

Имя входного файла: `polynomial.in`  
Имя выходного файла: `polynomial.out`  
Ограничение по времени: 2 секунды  
Ограничение по памяти: 256 мегабайт

Снежок нашел листок из старого учебника по математике. На нем перечислены  $t$  многочленов от двух переменных.

Теперь Снежок хочет их упростить. Помогите ему.

Во входном файле многочлены записаны в следующем формате:

- Многочлен не содержит пробелов и состоит только из арабских цифр и символов «n», «m», «+», «-», «\*», «^».
- Многочлен состоит из нескольких слагаемых.
- Если слагаемое входит со знаком минус, то перед ним пишется «-».
- Если слагаемое входит со знаком плюс и это не первое слагаемое, то перед ним пишется «+».
- Каждое слагаемое можно представить в виде  $a \cdot n^b \cdot m^c$ , где  $a$ ,  $b$  и  $c$  — целые неотрицательные числа (если не учитывать знак перед слагаемым), без ведущих нулей, не превышающие  $10^5$ .
- Слагаемое записывается в следующем формате:
  - Слагаемое является произведением чисел и степеней  $n$  и  $m$ .
  - Назовем числом последовательность подряд идущих цифр, слева и справа от которой стоят не цифры.
  - Перед каждым символом «^» находится переменная, то есть символ «n» или символ «m».
  - После каждого символа «^» находится число.
  - Таким образом, знак «^» обозначает, что переменная, стоящая перед ним, возводится в степень числа, стоящего после него. Назовем это множителем.
  - Если после символов «n» или «m» нет символа «^», то считается, что эта переменная в этом месте стоит в степени 1. Назовем каждую такую переменную множителем.
  - Все числа, которые не являются степенями (то есть, те, перед которыми не стоят символы «^»), назовем множителями.
  - По обе стороны от каждого символа «\*» находятся множители.
  - Слагаемое это произведение всех его множителей.
  - Обращаем ваше внимание, что некоторые соседние множители могут не разделяться символом «\*».

Для упрощенного многочлена должно выполняться следующее:

- Он записывается в таком же формате, как и исходный.
- Если многочлен тождественно равен нулю, нужно вывести «0».
- Если слагаемое представимо в виде  $a \cdot n^b \cdot m^c$ , то такое слагаемое должно быть записано как «an<sup>b</sup>m<sup>c</sup>», где на место «a», «b» и «c» подставлены соответствующие числа.
- Для каждой пары  $a$  и  $b$  существует не более одного слагаемого, представимого в виде  $k \cdot n^a \cdot m^b$ .
- Если  $n^a \cdot m^b$  входит в многочлен с коэффициентом 0, такое слагаемое писать не надо.
- Если  $n^a \cdot m^b$  входит в многочлен с коэффициентом, по модулю равным 1, в таком слагаемом не нужно писать коэффициент.

- В слагаемом вида  $a \cdot n^0 \cdot m^c$ , переменную «n» писать не надо.
- В слагаемом вида  $a \cdot n^1 \cdot m^c$ , после переменной «n» не нужно писать степень.
- Для переменной «m» тоже выполняются предыдущие два пункта.
- Слагаемые в многочлене могут идти в любом порядке.

## Формат входных данных

В первой строке задано целое число  $t$  — количество многочленов ( $1 \leq t \leq 100$ ). В следующих  $t$  строках заданы многочлены. Гарантируется, что все числа в них не превосходят  $10^5$ .

Длины всех строк не превосходят 100.

## Формат выходных данных

Для каждого многочлена выведите его в упрощенном виде.

Гарантируется, что в упрощенном многочлене все числа не превосходят  $10^9$ .

## Пример

| polynomial.in                                     | polynomial.out   |
|---------------------------------------------------|------------------|
| 5                                                 | $m - n + 2n^2m$  |
| $-n^2m^1 + n * nm3 + mn - n + m - n * m$          | $-2nm + 5n^2m^2$ |
| $n * n * m^0m^2 * 5 - m * 2n$                     | $23 - 100nm$     |
| $-100n * m + 23 + 0n^3$                           | 0                |
| $n - n$                                           | $nm^2$           |
| $100000 * 100000 * 100000 * n^3 * n^4 * 0 + nm^2$ |                  |

## Замечание

Пояснение к примеру:

$$-n^2 * m^1 + n * nm3 + mn - n + m - n * m = m - n + 2n^2m$$

$$n * n * m^0m^2 * 5 - m * 2n = -2nm + 5n^2m^2$$

$$-100n * m + 23 + 0n^3 = 23 - 100nm$$

$$n - n = 0$$

$$100000 * 100000 * 100000 * n^3 * n^4 * 0 + nm^2 = nm^2$$

Обращаем ваше внимание, что хотя в процессе вычисления значения коэффициента в первом слагаемом последнего многочлена оно и становится больше  $10^5$ , но в итоге это слагаемое равно  $0 \cdot n^3 \cdot n^4$ , значит для него ограничение на величину коэффициента тоже выполняется.

## Задача D. Макс и Дюк

Имя входного файла: `numpalindrome.in`  
Имя выходного файла: `numpalindrome.out`  
Ограничение по времени: 4 секунды  
Ограничение по памяти: 256 мегабайт

Макс и Дюк попали на мясокомбинат и нашли длинную-предлинную цепочку сосисок, которую можно представить как строку  $s$ , каждой сосиске соответствует маленькая латинская буква.

Они считают отрезок сосисок  $[a, b]$  вкусным, если непрерывная последовательность сосисок с  $a$  по  $b$  совпадает с этой же последовательностью в перевернутом виде. В каждый момент Макс наблюдает за отрезком сосисок  $[l, r]$  и задает Дюку странные запросы: сколько вкусных подотрезков видит Макс.

Вам необходимо узнать количество пар  $a$  и  $b$  таких, что  $l \leq a \leq b \leq r$  и отрезок  $[a, b]$  вкусный.

### Формат входных данных

В первой строке задано число  $n$  и  $m$  — длина строки  $s$  и количество запросов ( $1 \leq n, m \leq 500\,000$ ).

Во второй строке задана последовательность маленьких латинских букв длины  $n$  — строка  $s$ .

Далее следует  $m$  строк. В каждой записаны числа  $l$  и  $r$  — границы  $i$ -го запроса ( $1 \leq l \leq r \leq n$ ).

### Формат выходных данных

Для каждого запроса выведите количество вкусных подотрезков.

### Пример

| <code>numpalindrome.in</code> | <code>numpalindrome.out</code> |
|-------------------------------|--------------------------------|
| 8 5                           | 4                              |
| oobokkor                      | 12                             |
| 1 3                           | 1                              |
| 1 8                           | 6                              |
| 2 2                           | 5                              |
| 4 7                           |                                |
| 5 8                           |                                |

## Задача Е. Степенная башня Макса

Имя входного файла: `powerstower.in`  
Имя выходного файла: `powerstower.out`  
Ограничение по времени: 2 секунды  
Ограничение по памяти: 256 мегабайт

Макс — очень избалованный пёс. Недавно он выпросил у Кэти игрушечные кубики, на которых написаны числа. Он долго играл с ними, но вскоре ему это надоело. Поэтому он стал строить высокую башню из них. Сейчас в его башне  $n$  кубиков, на каждом из которых написано некоторое натуральное число  $a_i$ . Максу стало интересно, чему равно значение степенной башни  $a_1^{a_2^{a_3 \dots a_n}}$ . Но так как Макс пёс, то больших чисел удерживать в голове он не может, поэтому он хочет узнать результат по модулю 3. Помогите Макс.

Напомним, что операция возведение в степень правоассоциативна.

### Формат входных данных

В первой строке находится натуральное число  $n$  ( $1 \leq n \leq 10^5$ ). В следующей строке находится  $n$  натуральных чисел  $a_i$  ( $1 \leq a_i \leq 10^9$ ).

### Формат выходных данных

Выведите результат выражения  $a_1^{a_2^{a_3 \dots a_n}} \bmod 3$

### Примеры

| <code>powerstower.in</code> | <code>powerstower.out</code> |
|-----------------------------|------------------------------|
| 3<br>1 2 3                  | 1                            |
| 3<br>2 3 2                  | 2                            |

### Замечание

Во втором тестовом примере  $2^{3^2} = 512 \bmod 3 = 2$

## Задача F. Префиксы-суффиксы

Имя входного файла: `presuf.in`  
Имя выходного файла: `presuf.out`  
Ограничение по времени: 2 секунды  
Ограничение по памяти: 256 мегабайт

Гиджет считает, что Макссу нравятся только умные собаки, поэтому она решила почитать умных книжек. В одной из них она узнала понятия префикс и суффикс.

Префикс — это подстрока, начало которой совпадает с началом строки, а суффикс — это подстрока, конец которой совпадает с концом строки. Так, например, «ab» — один из префиксов строки «abacaba», а «25» — суффикс строки «ab125».

Гиджет заметила у хозяев в блокноте некоторую последовательность чисел, и ей стало интересно, существует ли хотя бы два, необязательно различных, числа таких, что один из префиксов первого равен одному из суффиксов второго.

К сожалению, эта задача оказалась для Гиджет слишком сложной. Помогите Гиджет в ее решении!

### Формат входных данных

В первой строке входного файла дано одно число  $n$  — количество чисел в блокноте. ( $1 \leq n \leq 1000$ )

Во второй строке записано  $n$  чисел  $a_i$  — числа из блокнота. ( $1 \leq a_i \leq 10^9$ )

### Формат выходных данных

В выходной файл выведите два числа  $x$  и  $y$  такие, что один из префиксов  $x$ -го числа в блокноте равен одному из суффиксов числа под номером  $y$ . Если таких двух чисел не существует, нужно вывести  $-1$ .

### Примеры

| <code>presuf.in</code> | <code>presuf.out</code> |
|------------------------|-------------------------|
| 5<br>11 12 13 14 15    | 3 1                     |
| 2<br>123456 123123     | 1 2                     |
| 1<br>1                 | 1 1                     |

## Задача G. Восстановление массива

Имя входного файла: `recovery.in`  
Имя выходного файла: `recovery.out`  
Ограничение по времени: 2 секунды  
Ограничение по памяти: 256 мегабайт

Кэти подарила Максy массив неубывающих чисел на день рождения. Он безумно ему понравился, однако друзья решили пошутить и добавили несколько цифр в некоторых числах массива, пока Макса не было рядом. Наш герой сразу заметил, что что-то не так — массив перестал быть неубывающим! Он впал в отчаяние, но на помощь ему пришла Гиджет — она сказала, что поможет ему восстановить массив.

Гиджет знает, что исходный массив возможно получить уже не получится, потому что его никто не помнит, но можно хотя бы попробовать восстановить свойство неубывания элементов, удаляя некоторые цифры из элементов массива. Конечно, изменять количество элементов массива не стоит — Макс сразу это заметит, поэтому удалять все цифры из какого-либо числа она не хочет. Так как Макс в полном отчаянии и времени у нее не так много, она хочет восстановить свойства исходного массива, удалив наименьшее количество цифр. Помогите ей с этой непростой задачей!

### Формат входных данных

В первой строке входного файла дано число  $n$  — количество элементов массива ( $1 \leq n \leq 1000$ ).

В следующих  $n$  строках даны элементы массива — целые неотрицательные числа  $a_i$ . Гарантируется, что числа в массиве не содержат нулей и суммарная длина всех чисел в массиве не превосходит 100 000.

### Формат выходных данных

В единственной строке выходного файла выведите минимальное количество цифр, которое надо удалить, чтобы элементы массива снова стали неубывающими.

Если не существует способа удалить несколько цифр так, чтобы в получившемся массиве числе неубывали, в единственной строке выходного файла выведите «-1».

### Примеры

| <code>recovery.in</code>  | <code>recovery.out</code> |
|---------------------------|---------------------------|
| 4<br>93<br>31<br>23<br>31 | 2                         |
| 4<br>93<br>31<br>23<br>12 | 3                         |
| 4<br>93<br>31<br>23<br>2  | -1                        |

### Замечание

В первом примере достаточно удалить первую цифру первого числа и вторую цифру второго, получится последовательность 3, 3, 23, 31, которая является неубывающей.

Во втором примере также нужно удалить первую цифру третьего числа, чтобы получить последовательность 3, 3, 3, 12.

## Задача Н. Расписание

Имя входного файла: `schedule.in`  
Имя выходного файла: `schedule.out`  
Ограничение по времени: 2 секунды  
Ограничение по памяти: 256 мегабайт

У хозяйки Макса Кэти очень много дел. Для удобства она решила пронумеровать все дела целыми неотрицательными числами в порядке убывания их важности (в частности дело с номером 0 самое важное).

Сейчас в распоряжении Кэти находятся  $n$  дней, в каждом из которых она выделила  $m$  моментов времени (по привычке дни и моменты Кэти также пронумеровала с нуля). Чтобы все успевать и при этом избегать рутины, девушка составила расписание, в котором решила придерживаться следующего правила. В  $i$ -ый день в момент времени  $j$  Кэти выбирает самое важное (минимальное по номеру) дело такое, которое она не делала в этот день ранее (то есть в моменты от 0 до  $j - 1$ ) и в прежние дни в  $j$ -ые моменты времени (в частности в нулевой день в нулевой момент Кэти будет занята делом 0).

Очень скоро Кэти поняла, что дела в таком расписании будут распределены единственным образом, а значит она сможет с легкостью узнавать, что ей необходимо сделать в каждый конкретный момент времени. Помогите ей в этом.

### Формат входных данных

В первой строке входного файла задано одно натуральное число  $k$  — количество запросов ( $1 \leq k \leq 10^5$ ).

В следующих  $k$  строках следуют сами запросы, каждый запрос — пара целых неотрицательных чисел  $(i, j)$ , где  $i$  — номер дня и  $j$  — номер момента ( $0 \leq i, j \leq 10^9$ ).

### Формат выходных данных

Для каждого запроса в отдельной строке выведите одно число — номер дела, которое необходимо выполнить в заданный момент времени.

### Примеры

| <code>schedule.in</code> | <code>schedule.out</code> |
|--------------------------|---------------------------|
| 6                        | 0                         |
| 1 1                      | 11                        |
| 2 9                      | 1                         |
| 3 2                      | 5                         |
| 3 6                      | 1                         |
| 5 4                      | 0                         |
| 8 8                      |                           |
| 4                        | 61                        |
| 25 36                    | 117                       |
| 59 78                    | 48                        |
| 87 103                   | 151                       |
| 100 243                  |                           |

## Задача I. Вентиляция

Имя входного файла: `treefirstedge.in`  
Имя выходного файла: `treefirstedge.out`  
Ограничение по времени: 3 секунды  
Ограничение по памяти: 256 мегабайт

Норман заблудился в вентиляции и уже четвёртую неделю ищет свою квартиру.

Вентиляция состоит из  $n$  узлов, соединённых  $n-1$  переходами таким образом, что между любыми двумя узлами существует ровно один путь.

Иногда Норман задаётся вопросом: в каком направлении идти, чтобы попасть в некоторый узел. Норман — всего лишь морская свинка, поэтому он не может запомнить все узлы и переходы между ними. Помогите ему узнать, куда идти.

### Формат входных данных

В первой строке входного файла задано число  $n$  — количество узлов в вентиляции ( $2 \leq n \leq 200\,000$ ).

В следующих  $n-1$  строках описаны переходы — по одному в строке. Каждый переход задаётся номерами узлов, которые он соединяет:  $a_i$  и  $b_i$  ( $1 \leq a_i, b_i \leq n$ ;  $a_i \neq b_i$ ). Гарантируется, что между любыми двумя узлами существует единственный путь по переходам.

В следующей строке задано число  $m$  — количество вопросов Нормана ( $1 \leq m \leq 100\,000$ ).

В следующих  $m$  строках описаны вопросы — по одному в строке. Каждый вопрос задаётся номером узла, в котором находится Норман ( $s_i$ ) и номером узла, куда он хочет попасть ( $t_i$ ) ( $1 \leq s_i, t_i \leq N$ ;  $s_i \neq t_i$ ).

Узлы нумеруются с 1.

### Формат выходных данных

Для каждого вопроса выведите номер узла, в который нужно идти из  $s_i$  напрямую, чтобы попасть в  $t_i$ . Обратите внимание, что ответ единственный, так как между любыми двумя вершинами существует ровно один путь.

### Пример

| treefirstedge.in | treefirstedge.out |
|------------------|-------------------|
| 5                | 3                 |
| 1 2              | 4                 |
| 1 3              | 1                 |
| 1 4              |                   |
| 3 5              |                   |
| 3                |                   |
| 5 2              |                   |
| 1 4              |                   |
| 4 3              |                   |

## Задача J. Треугольники

Имя входного файла: `triangles.in`  
Имя выходного файла: `triangles.out`  
Ограничение по времени: 2 секунды  
Ограничение по памяти: 256 мегабайт

Макс очень любит Гиджет! Поэтому очень хочет сделать ей подарок на День Рождения. Недолго думая, он решил подарить ей стеклянный куб — именно то, на что она намекала ему весь год.

Обойдя весь Манхэттен, Макс убедился, что стеклянные кубы нигде не продаются, но, к счастью, его приятель Дюк обнаружил у себя в кармане три стеклянных треугольника и с радостью поделился ими с Максом.

Теперь Макс достаточно собрать куб, разрезав имеющиеся у него стеклянные треугольники произвольное количество раз. Помогите Макс!

Если Макс удастся собрать куб, произвольным образом разрезав треугольники неограниченное количество раз и используя только прямолинейные разрезы, выведите максимально возможную длину ребра этого куба. Иначе, если не удастся, выведите «**Poor Max**».

### Формат входных данных

В каждой из трех строк входного файла содержится три числа  $A_i, B_i, C_i$  — длины сторон каждого из треугольников ( $1 \leq A_i, B_i, C_i \leq 10^6$ ). Гарантируется, что такие треугольники существуют и являются невырожденными.

### Формат выходных данных

Выведите ответ на задачу. Ответ будет считаться правильным, если он выведен с абсолютной или относительной погрешностью не более  $10^{-6}$ .

### Пример

| <code>triangles.in</code>   | <code>triangles.out</code> |
|-----------------------------|----------------------------|
| 3 4 5<br>5 12 13<br>7 24 25 | 4.4721359549995796         |