Содержание

Разминка

1 Задача A. A квадрат плюс Б квадрат [0.5 секунд, 256 mb]

Найдите количество решений уравнения вида $a^2 + b^2 = n$ в натуральных числах.

Формат входных данных

На первой строке число тестов t ($1 \leqslant t \leqslant 1000$).
Далее на каждой строке очередное число n_i ($1 \leqslant n_i \leqslant 10^9$).

Формат выходных данных

Для каждого теста выведите на отдельной строке число решений.

Примеры

sqrtab.in	sqrtab.out
4 1 2 5 1000	0 1 2 4

Замечание

$2 = 1^2 + 1^2$
$5 = 1^2 + 2^2 = 2^2 + 1^2$
$1000 = 10^2 + 30^2 = 30^2 + 10^2 = 18^2 + 26^2 = 26^2 + 18^2$

2 Задача B. Обратная перестановка [0.5 секунд, 256 mb]

Обратной перестановкой к перестановке p_1, p_2, ..., p_n называется перестановка q_1, q_2, ..., q_n, в которой q_i — это номер места, на котором стоит число i в перестановке p. К примеру, обратной к перестановке 1 4 2 3 является перестановка 1 3 4 2.

По данной перестановке длины n найдите обратную к ней перестановку.

Формат входных данных

В первой строке входного файла задано целое число n ($1 \leqslant n \leqslant 10$). Во второй строке заданы n целых чисел p_1, p_2, ..., p_n через пробел. Гарантируется, что эти числа образуют перестановку чисел 1, 2, ..., n.

Формат выходных данных

В первой строке выходного файла выведите n чисел через пробел — перестановку, обратную данной.

Примеры

inverse.in	inverse.out
1 1	1
2 1 2	1 2
4 1 4 2 3	1 3 4 2

3 Задача C. Лишние пробелы [0.5 секунд, 256 mb]

Дана строка. Напишите программу, которая удалит из этой строки все лишние пробелы. Пробел будем считать лишним, если:

1. он находится в самом начале строки, до самого первого слова;
2. он находится в конце строки, после самого последнего слова;
3. несколько пробелов расположены между двумя словами (проще говоря, если слова разделены более чем одним пробелом, тогда все пробелы кроме одного — лишние).

Формат входных данных

Во входном файле записана строка, длина которой не превышает 200 символов. Строка содержит только маленькие латинские буквы и пробелы.

Формат выходных данных

Выведите в выходной файл эту строку без лишних пробелов.

Примеры

spaces.in	spaces.out
first test	first test

Задачи

4 Задача D. Plus minus [0.5 секунд, 64 mb]

В каждой клетке поля $M \times N$ стоит либо плюс, либо минус. За один ход разрешается поменять знаки на противоположные в любом квадрате 2×2. Можно ли с помощью таких операций получить во всех клетках поля знаки плюс?

Формат входных данных

В первой строке числа M и N ($1 \leqslant N, M \leqslant 1000$). В следующих M строках содержится по N символов `+`, либо `-`.

Формат выходных данных

Ответ на вопрос задачи: слово `Yes` или `No`

Пример

plusminus.in	plusminus.out
3 3 -+- -+- ++-	No
3 3 -+- +++ -+-	Yes

5 Задача E. Поиск [1 секунда, 256 mb]

В этой задаче нужно уметь выяснять, содержится ли число в последовательности.

Формат входных данных

В первой строке входного файла заданы через пробел два целых числа n и k ($1 \leqslant n \leqslant 300\,000$, $1 \leqslant k \leqslant 300\,000$). Во второй строке задана последовательность из n отсортированных целых чисел a_1, a_2, ..., a_n, записанных через пробел ($1 \leqslant a_i \leqslant 10^9$). В третьей строке записаны запросы — k целых чисел b_1, b_2, ..., b_k записанных через пробел, в порядке возрастания ($1 \leqslant b_j \leqslant 10^9$).

Формат выходных данных

В выходной файл выведите k строк. В j-ой строке выведите "YES", если число b_j содержится в последовательности $\{a_i\}$, и "NO" в противном случае.

Примеры

find2.in	find2.out
3 3 2 3 5 1 2 3	NO YES YES
3 4 1 2 2 1 2 4 5	YES YES NO NO

6 Задача F. Тестирующая система [1 секунда, 256 mb]

Юный программист Саша написал свою первую тестирующую систему. Он так обрадовался тому, что она скомпилировалась, что решил пригласить школьных друзей на свой собственный контест.

Но в конце тура выяснилось, что система не умеет сортировать команды в таблице результатов. Помогите Саше реализовать эту сортировку.

Команды упорядочиваются по правилам ACM:

- по количеству решённых задач в порядке убывания;
- при равенстве количества решённых задач — по штрафному времени в порядке возрастания;
- при прочих равных — по номеру команды в порядке возрастания.

Формат входных данных

Первая строка содержит натуральное число n ($1 \leqslant n \leqslant 100\,000$) — количество команд, участвующих в контесте. В i-й из следующих n строк записано количество решенных задач S ($0 \leqslant S \leqslant 100$) и штрафное время T ($0 \leqslant T \leqslant 100\,000$) команды с номером i.

Формат выходных данных

В выходной файл выведите n чисел — номера команд в отсортированном порядке.

Пример

ejudge.in	ejudge.out
5 3 50 5 720 1 7 0 0 8 500	5 2 1 3 4

7 Задача G. Уставший профессор [1.5 секунд, 256 mb]

Уставший профессор вечером играет в увлекательную игру.

Изначально на доске слева направо записаны целые числа $a_1, a_2, \ldots, a_n$. Пока не уснет, профессор каждую секунду смотрит на числа, видит, что самое левое равно x, а самое правое равно y. Если x меньше, то профессор радуется, стирает слева x, а справа дописывает $(x+y) \mod 2^{30}$. Иначе профессор очень расстаивается, стирает y, а слева дописывает $(y-x) \mod 2^{30}$. Студенты подсчитали, что перед сном профессор успел сделать ровно k операций. Что было написано на доске, когда он наконец заснул? Для простоты можно считать, что доска в обе стороны бесконечна.

Формат входных данных

На первой строке n ($1 \leqslant n \leqslant 30\,000$) и k ($1 \leqslant k \leqslant 10^8$). На второй строке числа $a_1, a_2, \ldots, a_n$ ($0 \leqslant a_i < 10^9$) в порядке слева направо.

Формат выходных данных

На первой строке выведите все числа на доске после k операций. Выводить числа нужно в порядке слева направо.

Примеры

sleepgame.in	sleepgame.out
4 1 1 2 3 4	2 3 4 5
4 1000 1 2 3 4	1062488873 1072033429 1060433235 57573251
4 1 4 3 2 1	1073741821 4 3 2
4 2 4 3 2 1	5 1073741821 4 3

8 Задача H. Быстрое прибавление [4 секунды, 256 mb]

Есть массив целых чисел длины $n = 2^{24}$, изначально заполненных нулями. Вам нужно сперва обработать m случайных запросов вида "прибавление на отрезке" по модулю 2^{32}. Затем обработать q случайных запросов вида "сумма на отрезке" по модулю 2^{32}.

Формат входных данных

На первой строке числа m, q. ($1 \leqslant m, q \leqslant 2^{24}$). На второй строке пара целых чисел a, b от 1 до 10^9, используемая в генераторе случайных чисел.

```
1. unsigned int cur = 0; // беззнаковое 32-битное число
2. unsigned int nextRand() {
3.     cur = cur * a + b; // вычисляется с переполнениями
4.     return cur » 8; // число от 0 до 2^24 − 1.
5. }
```

Каждый запрос первого вида генерируется следующим образом:

```
1. add = nextRand(); // число, которое нужно прибавить
2. l = nextRand();
3. r = nextRand();
4. if (l > r) swap(l, r); // получили отрезок [l..r]
```

Каждый запрос второго вида генерируется следующим образом:

```
1. l = nextRand();
2. r = nextRand();
3. if (l > r) swap(l, r); // получили отрезок [l..r]
```

Сперва генерируются запросы первого вида, затем второго.

Формат выходных данных

Выведите сумму ответов на все запросы по модулю 2^{32}.

Примеры

fastadd.in	fastadd.out
5 5 13 239	811747796

9 Задача I. Линейная сумма [3 секунды, 256 mb]

Есть n случайных точек на прямой с координатами от 0 до $2^{32}-1$. У каждой точки есть значение от 0 до $2^{32}-1$. Вам нужно обработать q случайных запросов вида "сумма значений точек, с координатами от l до r включительно".

Формат входных данных

На первой строке числа n, q. ($1 \leqslant n \leqslant 2^{20}, 1 \leqslant q \leqslant 2^{23}$). На второй строке пара целых чисел a, b от 1 до 10^9, используемая в генераторе случайных чисел.

```
1. unsigned int cur = 0; // беззнаковое 32-битное число
2. unsigned int nextRand24() {
3.     cur = cur * a + b; // вычисляется с переполнениями
4.     return cur >> 8; // число от 0 до 2^24 - 1.
5. }
6. unsigned int nextRand32() {
7.     unsigned int a = nextRand24(), b = nextRand24();
8.     return (a << 8) ^ b; // число от 0 до 2^32 - 1.
9. }
```

Каждая точка генерируется следующим образом:

```
1. value = nextRand32(); // значение точки
2. x = nextRand32(); // координата точки
```

Каждый запрос генерируется следующим образом:

```
1. l = nextRand32();
2. r = nextRand32();
3. if (l > r) swap(l, r); // получили отрезок [l..r]
```

Сперва генерируются точки, затем запросы.

Формат выходных данных

Выведите сумму ответов на все запросы по модулю 2^{32}.

Примеры

linesum.in	linesum.out
5 5 13 239	3950632748

Замечание

```
   p = {value, x}
p[0] = {13, 41645}
p[1] = {7695587, 1253435649}
p[2] = {749170640, 2683600557}
p[3] = {2444595881, 1270561959}
p[4] = {3436107648, 486388002}
```

10 Задача J. Длинное выражение [2 секунды, 256 mb]

Выведите значение заданного арифметического выражения.

Формат входных данных

В первой строке входного файла задано выражение, состоящее из чисел, скобок и знаков бинарных операций. Каждое число в выражении это — целое неотрицательное число в промежутке от 0 до 10 000, включительно, записанное без ведущих нулей. Скобки бывают открывающие (‘(’) и закрывающие (‘)’). Операции задаются символами ‘+’, ‘-’, ‘*’ и ‘/’; знак умножения не может быть опущен. Гарантируется, что заданное выражение математически корректно, и результаты всех промежуточных операций — целые числа, не превышающие по модулю 10^9. Выражение не содержит каких-либо других символов, в частности, пробелов. Длина выражения не меньше 1 и не больше 1 000 000 символов.

Учтите, что операции с одинаковым приоритетом при отсутствии скобок выполняются слева направо. Например, выражение $a + b + c$ вычисляется как $(a + b) + c$.

Формат выходных данных

В первой строке выходного файла выведите одно число — значение заданного выражения.

Примеры

evalhard.in	evalhard.out
40-8/1*3	16
(5+50)/(2+3)	11

Бонус

11 Задача K. Дерево [3 секунды, 256 mb]

Задано подвешенное дерево, содержащее n ($1 \leqslant n \leqslant 1\ 000\ 000$) вершин. Каждая вершина покрашена в один из n цветов. Требуется для каждой вершины v вычислить количество различных цветов, встречающихся в поддереве с корнем v.

Формат входных данных

В первой строке входного файла задано число n. Последующие n строк описывают вершины, по одной в строке. Описание очередной вершины i имеет вид p_i c_i, где p_i — номер родителя вершины i, а c_i — цвет вершины i ($1 \leqslant c_i \leqslant n$). Для корня дерева $p_i = 0$.

Формат выходных данных

Выведите n чисел, обозначающих количества различных цветов в поддеревьях с корнями в вершинах $1, \ldots, n$.

Пример

tree.in	tree.out
5 2 1 3 2 0 3 3 3 2 1	1 2 3 1 1

12 Задача L. Все минимумы [0.5 секунд, 256 mb]

Внимание: в данной задаче принимаются только решения за $\mathcal{O}(n)$.

Дан массивы целых чисел $a_1, a_2, \ldots, a_n$.

Для каждого его подинтервала $[a_L \ldots a_R]$ определим $F(L, R) := min\{a_L, \ldots, a_R\}$.

Найдите

$$\sum_{1 \leqslant L \leqslant R \leqslant n} F(L, R)$$

то есть сумму минимумов всех подотрезков.

Внимание. Ваше решение должно иметь асимптотику $O(n)$.

Формат входных данных

Первая строка входных данных содержит натуральное число n ($1 \leqslant n \leqslant 100\,000$) — размер массива. Во второй строке через пробел заданы элементы массива, все числа целые от -10^6 до 10^6.

Формат выходных данных

Выведите единственное число — сумму минимумов всех подотрезков массива a.

Примеры

minsum.in	minsum.out
1 5	5
2 -10 1	-19
4 1 2 3 4	20
5 -3 2 -4 1 -5	-52

13 Задача M. Интересный разбор выражений [1 секунда, 256 mb]

Задача: дано арифметическое выражение, посчитайте значение.

В выражении присутствуют:

- Целые числа из диапазона $[-2^{31}, 2^{31})$.
- Операции:
  `+`, `-`, `*` (сложение, вычитание, умножение),
  `%` (остаток по модулю, не отрицателен),
  `/` (целочисленное деление, остаток неотрицателен),
  `^` (возведение в степень).
- Унарный минус.
- Скобки трех типов: `{ }` `[ ]` `( )`.
- Переменные с целочисленными значениями. Имена переменных — строки из букв латинского алфавита. Регистр важен.
- Функции `sqr` (квадрат числа), `cube` (куб числа), `sign` (знак числа).

Правила разбора выражения:

- Минус: после числа/имени или закрывающей скобки идет бинарный минус. Иначе минус унарный.
- Приоритеты операций: (`+`, `-`) затем (`*`, `%`, `/`) затем (`^`).
- Операции `+`, `-`, `*`, `/` левоассоциативны: `2-3+4 = (2-3)+4`.
- Операция возведения в степень `^` правоассоциативна: `3^3^3 = 3^27`.
- Если после имени идет открывающая скобка — это функция, иначе переменная.

Вычисление выражения: сперва происходит разбиение на лексемы и построение дерева вычислений, затем вычисляется значение выражения с помощью обхода дерево разбора слева направо. Сперва вычисляются значения аргументов операции в порядке слева направо, а когда все аргументы посчитаны, вычисляется значение оператора. Унарные операторы и функции в дереве имеют степень один, бинарные операторы имеют степень два.

Формат входных данных

Первые несколько строк содержат значения переменных в формате ⟨`name`⟩ = ⟨`value`⟩. Эти строки точно корректны. Имена переменных в присваиваниях могут совпадать, используется последнее значение. Последняя строка содержит арифметическое выражение, значение которого нужно посчитать. Арифметическое выражение может содержать синтаксические ошибки, и ошибки, непозволяющие вычислить его значение. Подробнее читайте ниже. Суммарная длина всех строк входного не более 10^6. Во входном файлы используются только допустимые символы.

Формат выходных данных

Если арифметическое выражение некорректно, выведите `Error:` ⟨`ошибка`⟩. Если произошло несколько ошибок, нужно выводить только первую.

Ошибки на этапе разбора выражения на лексемы в порядке приоритета:

- `unmatched bracket` — лишние или не парные скобки.
- `parsing expression` — численные значения и операции не чередуются.
- `undefined name` — имя, которое не соответствует ни переменной, ни функции.
- `too long integer` — используется число не из диапазона $[-2^{31}, 2^{31})$.

Ошибки на этапе вычисления значения выражения, выражение вычисляется обходом дерева слева направо, нас интересует первая ошибка именно в этом порядке:

- `integer overflow` — конечное, или одно из промежуточных значений лежат за пределами диапазона $[-2^{31}, 2^{31})$.
- `dividing by zero` — деление на ноль.
- `negative power is not allowed` — возведение в отрицательную степень.

Если арифметическое выражение задано корректно и может быть корректно вычислено, выведите целое число — значение выражения.

Примеры

exprf.in	exprf.out
x = 2 value = 3 x^value - cube(2+3)	-117
-2+3-(-2+3)+[-100]-{-100}--100+ sqr(-2)+sign(-1)+-2^5	71
x = 2 x^100	Error: integer overflow
()	Error: parsing expression
(2 2 / 0 + 1000000000000	Error: unmatched bracket
(2 2) / 0 + 1000000000000	Error: parsing expression
(2 + x) / 0 + 1000000000000	Error: undefined name
(2 + 2) / 0 + 1000000000000	Error: too long integer
(2 + 2) / 0	Error: dividing by zero
2^(2 - sqr(2))	Error: negative power is not allowed
sqr = 8 sqr = 10 sqr + (sqr + 2 + sqr) + sqr(sqr(sqr)) + sqr	10042