# Содержание



---

Обратите внимание, входные данные лежат в **стандартном потоке ввода** (он же stdin), вывести ответ нужно в **стандартный поток вывода** (он же stdout).

Обратите внимание на `GNU C++` компиляторы с суффиксом `inc`.
Подни можно пользоваться дополнительной библиотекой (`optimization.h`).
То есть, использовать быстрый ввод-вывод: **пример про числа и строки**.
И быструю аллокацию памяти (ускоряет vector-set-map-весь-STL): пример.

Для тех, кто хочет разобраться, как всё это работает.
Короткая версия быстрого ввода-вывода (тык) и короткая версия аллокатора (тык).

# Must have

## Задача 3A. Обмен [0.25 sec, 256 mb]

Пусть все натуральные числа исходно организованы в список в естественном порядке. Разрешается выполнить следующую операцию: $swap(a, b)$. Эта операция возвращает в качестве результата расстояние в текущем списке между числами $a$ и $b$ и меняет их местами.

Задана последовательность операций *swap*. Требуется вывести в выходной файл результат всех этих операций.

### Формат входных данных

Первая строка входного файла содержит число $n$ ($1 \leqslant n \leqslant 200\,000$) — количество операций. Каждая из следующих $n$ строк содержит по два числа в диапазоне от 1 до $10^9$ — аргументы операций *swap*.

### Формат выходных данных

Для каждой операции во входном файле выведите ее результат.

### Пример

| stdin | stdout |
|---|---|
| 4<br>1 4<br>1 3<br>4 5<br>1 4 | 3<br>1<br>4<br>2 |

### Замечание

Хеш-таблицы из `STL` (`unordered_map` и т.д.) получат `TL`.

Придётся писать свою хеш-таблицу.

Самая быстрая и простая хеш-таблица – хеш-таблица с открытой адресацией.

## Задача 3B. Мышата [0.5 sec, 256 mb]

У мышонка сегодня день рождения! По этому поводу он решил собрать друзей. Друзья приходят к мышонку и садятся вокруг круглого стола.

Вначале за столом сидит один мышонок, празднующий день рождения. Все его друзья приходят по одному и садятся слева или справа от кого-то из уже сидящих за столом.

Определите, в каком порядке будут сидеть друзья мышонка, когда все придут.

### Формат входных данных

В первой строке записано имя мышонка, празднующего день рождения, и одно число $n$ — количество его друзей ($1 \leqslant n \leqslant 100\,000$). В каждой из следующих $n$ строк записаны имя $a$ очередного приходящего гостя, имя $b$ того, рядом с кем он садится, и буква «`r`», если $a$ садится справа от $b$, или буква «`l`», если слева.

Все имена будут различными строками из латинских букв длиной от 1 до 10 символов. Строчные и заглавные буквы следует считать различными. Гарантируется, что тот, рядом с кем садится очередной гость, уже сидит за столом.

### Формат выходных данных

Выведите $n + 1$ строку, по одному имени в каждой строке. Выводите сидящих за столом в порядке движения вправо, начиная с мышонка.

### Пример

| stdin | stdout |
|---|---|
| Jerry 4<br>Tuffy Jerry l<br>Tom Jerry r<br>Spike Tom l<br>Quacker Jerry r | Jerry<br>Quacker<br>Spike<br>Tom<br>Tuffy |

### Подсказка по решению

Чтобы для строки хранить число, её номер, можно воспользоваться `map<string, int>`.

# Обязательные задачи

## Задача 3C. Сложение и вычитание [0.2 sec, 256 mb]

Выведите значение заданного арифметического выражения, состоящего из чисел, скобок и знаков сложения и вычитания.

### 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 ‘-’. Гарантируется, что заданное выражение математически корректно, и результаты всех промежуточных операций — целые числа, не превышающие по модулю 10 000. Выражение не содержит каких-либо других символов, в частности, пробелов. Длина выражения не меньше 1 и не больше 1000 символов.

Учтите, что операции 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

| stdin | stdout |
|---|---|
| 48-13 | 35 |
| 5-(52+3) | -50 |

### Замечание

Мы подробно изучали это неделю назад на практике. Полезно глянуть разбор.

## Задача 3D. Длинное выражение [0.2 sec, 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

| stdin | stdout |
|---|---|
| 40-8/1*3 | 16 |
| (5+50)/(2+3) | 11 |

### Замечание

Мы подробно изучали это неделю назад на практике. Полезно глянуть разбор.

## Задача 3E. Хороша ли подстрока? [0.25 sec, 256 mb]

Вам дана строка из круглых скобок. Нужно отвечать на запросы вида “есть ли у подстроки $[l..r]$ исходной строки циклический сдвиг, являющийся правильной скобочной последовательностью (*ПСП*)?”.

Напомним определение *ПСП*:

- Пустая строка – ПСП.
- Если $s$ – ПСП, то ‘$(s)$‘ – тоже.
- Если $s_1, s_2$ – ПСП, то ‘$s_1 s_2$‘ – тоже.

### Формат входных данных

Строка $s$ из круглых скобок длины от 1 до $10^6$.

На второй строке количество запросов $q$ ($1 \leqslant q \leqslant 10^5$).

Следующие $q$ строк содержат пары чисел $l$, $r$ ($1 \leqslant l \leqslant r \leqslant |s|$).

### Формат выходных данных

Для каждого запроса, если ответ да, выведите `1`, иначе `0`.

### Примеры

| stdin | stdout |
|---|---|
| ()())(<br>6<br>1 2<br>1 4<br>5 6<br>3 6<br>1 5<br>2 5 | 111100 |

### Замечание

Это простая задача на понимание ПСП.

Запрещено пользоваться высокоуровневыми заклинаниями типа «дерево отрезков!»

## Задача 3F. Художник [1.5 sec, 256 mb]

Енот Вася — начинающий художник. Недавно он приобрёл подержанную кисточку (как рассказал продавец, она сделана из хвоста самого Малевича!), достал холст и решил произвести на свет свой первый шедевр.

Как оказалось, лет кисточке немало, потому она способна лишь ставить кляксы в форме пяти квадратиков, расположенных крестиком (координаты их центров будут равны $(x, y)$, $(x-1, y)$, $(x, y-1)$, $(x+1, y)$, $(x, y+1)$). Вася поставил $N$ клякс, разочаровался в идее первого шедевра и задумался о месте для нового. Но ведь если он закрасил весь холст, писать будет негде...

Выясните, закрасил ли своими действиями Вася весь холст.

### Формат входных данных

Первая строка входного файла содержит три целых числа $W$, $H$ и $N$ — ширину и высоту холста в квадратиках ($1 \leqslant W, H \leqslant 10^9$) и количество клякс, поставленных Васей ($0 \leqslant N \leqslant 1\,000\,000$). Следующие $N$ строк содержат по два целых числа $x_i, y_i$ каждая — координаты среднего квадратика $i$-й кляксы ($-10^9 \leqslant x_i, y_i \leqslant 10^9$). Клетка $(x, y)$ находится на холсте, если $1 \leqslant x \leqslant W$ и $1 \leqslant y \leqslant H$. Части клякс, оказавшиеся вне холста, не учитываются.

### Формат выходных данных

Выведите «`Yes`», если Вася закрасил весь холст, и «`No`» в противном случае.

### Пример

| stdin | stdout |
|---|---|
| 2 2 2<br>1 1<br>2 2 | Yes |

### Подсказка по решению

Может пригодиться хеш-таблица.

Может пригодиться мозг.

## Задача 3G. Быстрое прибавление [5 sec, 256 mb]

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

### 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.

```
0. uint32_t a, b; // даны во входных данных
1. uint32_t cur = 0; // беззнаковое 32-битное число
2. uint32_t 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 второго типа по модулю $2^{32}$.

### Примеры

| stdin | stdout |
|---|---|
| 5 5<br>13 239 | 811747796 |

### Замечание

Последовательность запросов в тесте из примера:

```
[13..170] += 0
[28886..375523] += 2221
[2940943..13131777] += 4881801
[2025901..10480279] += 4677840
[4943766..6833065] += 9559505
get sum [13412991..13937319]
get sum [1871500..6596736]
get sum [7552290..14293694]
get sum [1268651..16492476]
get sum [2210673..13075602]
```

### Подсказка по решению

Есть простое решение за линейное время.

Запрещено пользоваться высокоуровневыми заклинаниями типа «дерево отрезков!»

# Дополнительные задачи

## Задача 3H. Минимумы в подматрицах [0.4 sec, 256 mb]

Дана матрица $n \times n$, состоящая из целых чисел. Для каждой её подматрицы размера $L \times L$ найдите минимум в этой подматрице. Подматрицей здесь называется "подпрямоугольник".

**Внимание.** Решение должно работать за $O(n^2)$.

### Формат входных данных

Первая строка входных данных содержит два целых числа $n$ и $L$ ($1 \leqslant L \leqslant n \leqslant 1000$). Далее в $n$ строках идет описание матрицы $n \times n$, по $n$ чисел в каждой строке. Все числа в матрицы целые, от $-10^9$ до $10^9$.

### Формат выходных данных

Выведите $n - L + 1$ строку по $n - L + 1$ числу в каждой. $j$-е число в $i$-й строке должно быть равно минимуму в подматрице размера $L \times L$ с левым верхнем углом на пересечении $i$-й строки и $j$-го столбца исходной матрицы.

### Примеры

| stdin | stdout |
|---|---|
| 1 1<br>5 | 5 |
| 2 1<br>2 1<br>3 4 | 2 1<br>3 4 |
| 2 2<br>2 1<br>3 4 | 1 |
| 4 2<br>4 5 3 2<br>1 2 5 4<br>3 4 2 3<br>1 3 5 5 | 1 2 2<br>1 2 2<br>1 2 2 |

### Подсказка по решению

Есть простое решение за $\mathcal{O}(1)$ на запрос.

## Задача 3l. Game [1.5 sec, 256 mb]

Вася и Петя снова затеяли очень интересную игру! Петя выписывает на доску по порядку $n$ чисел. Теперь мальчики, не советуясь, выбирают по 2 числа от 1 до $n$.

Пусть Петя выбрал числа $l$ и $r$ ($l \leqslant r$). Тогда очки, заработанные им, вычисляются по формуле $\sum\limits_{k=l}^{r} b_k \cdot \min\limits_{k=l..r} b_k$, где $b_k$ — $k$-ое число на доске. Аналогично вычисляются очки, заработанные Васей. Затем мальчики говорят друг другу свои очки. У кого очков меньше, тот проиграл. Если очков одинаковое количество, то это считается ничьёй.

Представьте, что вы — Вася. И вы очень хотите хотя бы не проиграть. При этом очень не хотите быть пойманными на жульничестве (то есть хотите, чтобы названное вами число очков возможно было получить честным способом).

Напишите программу, которая находит отрезок, дающий максимальное количество очков.

### Формат входных данных

В первой строке ввода задано натуральное число $n$ ($1 \leqslant n \leqslant 2\,000\,000$) — количество чисел на доске. В следующей строке содержатся $n$ целых чисел $b_k$ через пробел в том же порядке, что они написаны на доске ($|b_k| \leqslant 100\,000$).

### Формат выходных данных

Выведите число, которое должен назвать Вася, чтобы заведомо не проиграть и не быть пойманным на жульничестве. Во второй строке выведите границы отрезка, чтобы Вася смог доказать свою честность, если его начнут подозревать в жульничестве. Удачи!

### Пример

| stdin | stdout |
|---|---|
| 3<br>1 2 3 | 10<br>2 3 |

### Замечание

Отрицтальные числа! Это важно. Стек в помощь.

## Задача 3J. Интересный разбор выражений [2 sec, 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

| stdin | stdout |
|---|---|
| x = 2<br>value = 3<br>x^value - cube(2+3) | -117 |
| -2+3-(-2+3)+[-100]-{-100}--100+<br>sqr(-2)+sign(-1)+-2^5 | 71 |
| x = 2<br>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<br>sqr = 10<br>sqr + (sqr + 2 + sqr) +<br>sqr(sqr(sqr)) + sqr | 10042 |

### Замечание

Тесты к задаче устроены так: изначально тест 1. Затем каждый студент, сдавший задачу, имеет право добавить пару тестов. Чтобы сдать задачу в этом году нужно пройти всё, что добавили ваши предшественники.