Задача 1. Разница множеств

Источник:	базовая*
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

Дано два массива целых чисел A и B.

Требуется найти все такие значения элементов массива A, которых нет среди элементов массива B.

Замечание: В задаче необходимо использовать функцию `qsort` из стандартной библиотеки языка C.

Формат входных данных

В первой строке записано целое число N ($1 \leqslant N \leqslant 10^5$) — количество элементов массива A.

Во второй строке через пробел записано N целых чисел, каждое из которых не превосходит 10^9 по абсолютной величине — элементы массива A.

В следующих двух строках в аналогичном формате записаны элементы массива B.

Формат выходных данных

В первой строке выведите одно целое число — количество значений, удовлетворяющих описанному условию.

Во второй строке выведите все такие значения в порядке возрастания.

Примеры

input.txt	output.txt
7 1 2 3 3 6 8 8 4 1 3 7 9	3 2 6 8
3 1 2 3 3 3 2 1	0

Задача 2. Растущий массив

Источник:	базовая*
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

В данной задаче нужно реализовать массив переменного размера, в который можно дописывать элементы, не зная заранее его окончательный размер. Используя эту структуру данных, нужно решить приведённую ниже задачу.

В первой строке записано целое число N — количество записей ($1 \leqslant N \leqslant 2 \cdot 10^5$). В остальных N строках содержатся записи, по одной в строке.

Для каждой записи указаны ключ и значение через пробел. Ключ — это целое число в диапазоне от 0 до 10^6 включительно, а значение — это строка от одного до семи символов включительно, состоящая только из маленьких букв латинского алфавита.

Требуется вывести ровно те же самые N записей, но в другом порядке. Записи должны быть упорядочены по возрастанию ключа. Если у нескольких записей ключ равный, то нужно упорядочить их в том порядке, в котором они встречаются по входном файле.

Важно: Решать задачу **нужно** следующим образом (другие решения засчитываться **не** будут). Нужно завести 10^6 **массивов** переменного размера, и в каждый k-ый массив складывать все записи с ключом, равным k. После раскидывания записей по массивам достаточно будет пробежаться по массивам в порядке увеличения k и распечатать их.

Пример

input.txt	output.txt
7 3 qwerty 3 string 6 good 1 a 3 ab 2 hello 5 world	1 a 2 hello 3 qwerty 3 string 3 ab 5 world 6 good

Пояснение к примеру

В примере 7 записей с ключами 1, 2, 3, 5 и 6 — именно в таком порядке записи и выведены в выходном файле. Обратите внимание, что есть три записи с ключом 3: `qwerty`, `string`, `ab`. Они выведены ровно в том порядке, в котором они идут во входном файле.

Задача 3. Тасовка перфокарт

Источник:	базовая*
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	3 секунды
Ограничение по памяти:	**специальное**

Имеется две колоды перфокарт: левая колода и правая. Изначально в каждой колоде ровно N перфокарт. В левой колоде перфокарты пронумерованы числами от 1 до N по порядку, если просматривать их сверху вниз. В правой колоде перфокарты пронумерованы числами от -1 до $-N$ по порядку, если просматривать их сверху вниз.

Для перемешивания колод нужно выполнить M заданных операций, каждая операция заключается в перекладывании одной карты. Каждая операция обозначается одной шестнадцатеричной цифрой в диапазоне от `0` до `F` (15) включительно. Операция определяется значениями четырёх битов в двоичной записи этой цифры:

- Если старший бит (8) единичный, то нужно взять карту с правой колоды, а иначе — с левой колоды.
- Если предпоследний бит (4) единичный, то нужно взять карту снизу колоды, а иначе — сверху колоды.
- Если второй бит (2) единичный, то нужно положить карту в правую колоды, а иначе — в левую.
- Если младший бит (1) единичный, то нужно положить карту в колоду снизу, а иначе — сверху.

Если в какой-то момент нужно выполнить операцию, которая предписывает взять карту из пустой колоды, то такую операцию нужно пропустить (ничего не делая).

В первой строке записано два целых числа: N — начальное количество карт в каждой колоде и M — сколько операций нужно выполнить ($1 \leqslant N \leqslant 5 \cdot 10^5$, $0 \leqslant M \leqslant 5 \cdot 10^6$).

Во второй строке записано подряд ровно M символов — описание операций в порядке их выполнения. Каждый символ является шестнадцатеричной цифрой и изменяется в диапазоне от `0` до `9` или от `A` до `F` включительно.

После выполнения всех операций требуется вывести содержимое левой колоды в первой строке выходного файла, и содержимое правой колоды — во второй строке. В каждой строке нужно сначала вывести целое число K — количество карт в колоде после выполнения всех операций, а затем через пробел K целых чисел — номера перфокарт в колоде, перечисленные в порядке сверху вниз.

Важно: Требуется хранить каждую колоду в **кольцевом буфере** размером ровно на $(2N + 1)$ элементов. Память под кольцевые буферы выделяйте динамически.

Пример

input.txt	output.txt
5 0	5 1 2 3 4 5 5 -1 -2 -3 -4 -5
5 10 180FA45DB2	6 -1 2 3 4 5 -5 4 1 -3 -4 -2
3 20 CCCCCCCCCC9999999999	6 -1 -2 -3 1 2 3 0

P.S. По второму примеру можно заметить, что команды `0`, `F`, `A` и `5` никогда ничего не изменяют.

Задача 4. Вычисления на стеке

Источник:	основная
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

В обратной польской записи математических выражений операнды записываются перед знаками операций. Например, выражение “3 + 4” в обратной нотации будет записываться как “3 4 +”. Это позволяет не писать скобки и вообще не определять приоритет операций, а вычислять выражение, читая его слева направо. При таком чтении, если мы встречаем число, то кладём его в стек. Если встречаем операцию, то достаём её операнды из стека, применяем операцию к ним, и результат возвращаем в стек. В данной задаче вам необходимо реализовать стековую машину, которая по обратной польской записи вычисляет значение выражения. Все операции должны выполняться в кольце вычетов по модулю $10^9 + 7$. Например, если бы операции выполнялись в кольце вычетов по модулю 5, то результатом сложения $3 + 4 = 7, 7 \, mod \, 5 = 2$ — была бы двойка.

Формат входных данных

В первой строке записано целое число N — количество слов ($1 \leqslant N \leqslant 10^5$). Во второй строке N слов, разделенных пробелами. Каждое слово представляет из себя либо одну из операций ($+$, $-$, $*$), либо целое число из кольца вычетов. Каждая операция принимает ровно два операнда. Гарантируется, что количество операций на единицу меньше количества чисел. Также гарантируется, что когда приходит операция, в стеке есть как минимум два числа. Другими словами, входные данные всегда корректные.

Формат выходных данных

Нужно вывести одно целое число — результат вычисления выражения.

Пример

input.txt	output.txt
11 5 5 * 3 3 * 1 1 * + -	15

Пояснение к примеру

В привычной (инфиксной) нотации выражение из примера выглядело бы, как $5 * 5 - (3 * 3 + 1 * 1)$.

Задача 5. Вычисление синуса

Источник:	основная*
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

В данной задаче нужно научиться вычислять синус. Использовать функцию `sin` из стандартной библиотеки или откуда-то ещё **запрещено**.

Подсказка: используйте ряд Тейлора.

Формат входных данных

В первой строке записано одно целое число N — количество аргументов, для которых нужно вычислить синус ($1 \leqslant N \leqslant 10^5$). Далее идёт N строк, по одному вещественному числу X в каждой. Каждое число — это число, синус которого надо вычислить.

Все числа X по абсолютной величине не превышают единицу (заданы в радианах).

Формат выходных данных

Выведите N строк, в каждой строке одно вещественное число, которое равно $\sin X$ для соответствующего аргумента X из входного файла.

Рекомендуется выводить числа с помощью формата `"%0.15lf"`, чтобы выводилось 15 знаков после десятичной точки. Ошибка в каждом вашем ответе не должна превышать 10^{-12}.

Пример

input.txt	output.txt
5 0.0 0.523598775598298 0.785398163397448 1.047197551196597 1.570796326794896	0.000000000000000 0.500000000000000 0.707106781186548 0.866025403784439 1.000000000000000

Задача 6. Подсчёт недавних событий

Источник:	основная
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

Когда на сервер приходит запрос, в лог добавляется запись, которая имеет вид: [$time$], где $time$ — время, когда пришел запрос в миллисекундах. Каждый раз, когда приходит новый запрос, нам необходимо ответить, сколько запросов пришло на сервер за последние M миллисекунд.

Формат входных данных

В первой строке через пробел записано два целых числа: N ($1 \leqslant N \leqslant 10^5$) — сколько всего записей о запросах в логе и M ($1 \leqslant M \leqslant 10^9$) — за какой интервал в миллисекундах проверяем количество запросов. В следующих N строках записаны записи лога. Каждая запись представляет из себя целое число ($0 \leqslant time_i \leqslant 10^9$). Числа идут в порядке неубывания.

Формат выходных данных

Вывести N целых чисел — количество запросов, которое зарегистрировал сервер за последние M миллисекунд до момента i-ого запроса.

Пример

input.txt	output.txt
10 1000 1 100 100 100 1001 1002 1002 1005 3000 5000	1 2 3 4 5 5 6 7 1 1

Задача 7. Стек с запросами

Источник:	основная
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

Необходимо реализовать стек, который поддерживает стандартные для стека операции:
- **push(int val)** — добавляет число в стек,
- **int pop()** — достает число из стека.

Кроме того, нужно также уметь отвечать на запросы **min** и **max**:
- **int min()** — возвращает минимум из тех чисел, которые находятся в стеке.
- **int max()** — возвращает максимум из тех чисел, которые находятся в стеке.

min и **max** должны работать за O(1), т.е. мы должны хранить заготовленный ответ на любой из этих запросов, чтобы каждый раз не просматривать содержимое стека.

Формат входных данных

В первой строке записано целое число N — количество операций со стеком ($1 \leqslant N \leqslant 10^6$). В следующих N строках записаны операции (в случае с **push** через пробел записан операнд — целое число, которое умещается в int).

Гарантируется, что при выполнении запросов **min**, **max** и **pop** стек содержит хотя бы один элемент.

Формат выходных данных

Необходимо, чтобы в выходном файле на отдельных строчках лежали целые числа — возвращаемые значения операций **pop**, **min**, **max** в том же порядке, в котором они вызываются.

Пример

input.txt	output.txt
8 push 1 push 2 push 3 min max pop pop pop	1 3 3 2 1

Задача 8. Очередь с запросами

Источник:	Повышенной сложности
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

Необходимо реализовать очередь, которая поддерживает стандартные для очереди операции:

- **enqueue(int val)** — добавляет число в очередь,
- **int dequeue()** — достает число из очереди.

Кроме того, нужно также уметь отвечать на запросы **min** и **max**:

- **int min()** — возвращает минимум из тех чисел, которые находятся в очереди.
- **int max()** — возвращает максимум из тех чисел, которые находятся в очереди.

min и **max** должны работать за O(1), т.е. мы не должны каждый раз просматривать содержимое очереди.

Формат входных данных

В первой строке записано целое число N — количество операций с очередью ($1 \leqslant N \leqslant 10^6$). В следующих N строках записаны операции (в случае с **enqueue** через пробел записан операнд — целое число, которое умещается в int).

Гарантируется, что при выполнении запросов **min**, **max** и **pop** очередь содержит хотя бы один элемент.

Формат выходных данных

Необходимо, чтобы в выходном файле на отдельных строчках лежали целые числа — возвращаемые значения операций **dequeue**, **min**, **max** в том же порядке, в котором они вызываются.

Пример

input.txt	output.txt
8 enqueue 1 enqueue 2 enqueue 3 min max dequeue dequeue dequeue	1 3 1 2 3

Задача 9. Скобки

Источник:	повышенной сложности
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	2 секунды
Ограничение по памяти:	разумное

Как многим известно, основу синтаксиса LISP-подобных языков составляют скобочки. Круглые, фигурные, квадратные — на любой вкус. Конечно, там применяют и другие символы, но главное — скобочки.

В лабораториях Ну Гляди-какого-умного Университета (НГУ) придумали новый язык программирования — Uncommon LISP. Он впитал в себя самую суть всех функциональных языков. В нём кроме скобочек нет ничего...

В первой версии языка корректной программой на Uncommon LISP считалась правильная скобочная последовательность из четырех видов скобок: (), {}, `[]` и <>. Правильная скобочная последовательность определяется следующим образом:

- пустая строка — правильная скобочная последовательность;
- правильная скобочная последовательность, взятая в скобки одного типа — правильная скобочная последовательность;
- правильная скобочная последовательность, к которой приписана слева или справа правильная скобочная последовательность — тоже правильная скобочная последовательность.

К сожалению, эти правила были слишком сложными, так что во второй версии языка (Uncommon LISP v2) корректными программами стали считаться также те программы, из которых можно получить правильную скобочную последовательность, переставляя скобки определённым образом. В последовательности можно переставлять местами скобки, стоящие рядом, если они обе открывающие или обе закрывающие. Такую операцию можно выполнять сколько угодно раз, в том числе по несколько раз переставляя одни и те же скобки.

Необходимо определить для данной скобочной последовательности, является ли она корректной программой для Uncommon LISP v2.

Формат входных данных

В первой строке входного файла записано целое число N — количество тестов, следующих ниже. Каждый тест — это непустая строка, состоящая из скобок. Суммарная длина всех строк не превышает 1 000 000.

Формат выходных данных

В выходной файл необходимо для каждого теста в отдельную строку вывести ответ `T`, если заданную скобочную последовательность можно привести к правильному виду, и `NIL`, если нельзя.

Пример

input.txt	output.txt
2 ([)])([]	T NIL