Данный набор ориентирован на знакомство со стандратной библиотекой самого лучшего языка программирования C++.

Условия задач содержат лишь краткие рекомендации по выполнению заданий и вам придётся обратиться к полной документации библиотеки. Один из самый удобный способов — это воспользоваться сайтом `https://cppreference.com`.

Для подключения компонент стандартной библиотеки не забывайте добавлять соответствующий `#include`, например:

```
#include <vector>
```

для использования класса `vector`.

При использовании компилятора G++ (и его производных, например, mingw под windows), можно подключить разом почти всю стандартную библиотеку:

```
#include <bits/stdc++.h>
```

Так же в языке C++ использует понятие пространства имён. Нам же в ближайшее время оно не понадобится, поэтому, для более удобного использования, сразу после всех подключений удобно добавить строку:

```
using namespace std;
```

Если вы незаслуженно получаете вердикт `Time Limit Exceeded` в системе тестирования, то добавьте следующие волшебные строки в начало функции `main`:

```
ios_base::sync_with_stdio(false);
cin.tie(NULL);
```

Подробнее можно прочитать об этом здесь: `https://codeforces.com/blog/entry/5217?locale=ru` — прочитайте внимательно, т.к. данные конструкции влияют на работу ввода-вывода и могут привести к странному поведению в программе при неправильном их использовании.

Таким образом, минимальный шаблон для написания решения к первой задаче будет выглядеть:

```
#include <vector>
#include <algorithm>
using namespace std;

int main() {
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);

    // код решения
}
```

Задача 1. Мой первый вектор

Имя входного файла:	`стандартный поток ввода`
Имя выходного файла:	`стандартный поток вывода`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Дана последовательность из N чисел. Требуется вывести её в обратном порядке и в отсортированном порядке.

Во входных данных не задано ограничение на длину последовательности. Но оно вам и не важно, если использовать вектор:

```
#include <vector>
using namespace std;

// много вашего кода

int n; cin >> n;
vector<int> v(n, 0);
```

Для разворота контейнера вы можете попробовать функцию `reverse` из стандартной библиотеки c++:

```
#include <algorithm>

// много вашего кода

reverse(v.begin(), v.end());
```

Аналогично, для сортировки рекомендуется использовать функцию `sort`.

Формат входных данных

В первой строке файла записано целое положительное число N. Во второй строке через пробел записаны N целых чисел a_i ($0 \leqslant a_i \leqslant 10^9$).

Формат выходных данных

В первой строке выведите заданную последовательность чисел в обратном порядке.

Во первой строке выведите заданную последовательность чисел в отсортированном по неубыванию порядке.

Примеры

стандартный поток ввода	стандартный поток вывода
5 1 1 12 10 5	5 10 12 1 1 1 1 5 10 12

Замечания

Описание контейнер `vector`: `https://en.cppreference.com/w/cpp/container/vector`.

Описание функции `reverse`: `https://en.cppreference.com/w/cpp/algorithm/reverse`.

Описание функции `sort`: `https://en.cppreference.com/w/cpp/algorithm/sort`.

Задача 2. Вектор в векторе

Имя входного файла:	`стандартный поток ввода`
Имя выходного файла:	`стандартный поток вывода`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Требуется выполнить M операций над N динамическими массивами.

Операции бывают двух типов:

1. `push v u` — добавить в конец вектора под номером v (нумерация веторов задана с нуля) целое число u.
2. `pop v` — удалить с конца вектора под номером v последний элемент.

Для решения этой задачи по-прежнему подойдёт контейнер `std::vector` из стандартной библиотеки. Но, в отличие от предыдущий задачи, здесь придётся оперировать двумерным динамическим массивом:

```
vector<vector<int>> v;
int n; cin >> n;
v.resize(n);
```

Формат входных данных

В первой строке файла задано два целых положительных числа N и M.

Далее M строках заданы операции над векторами, по одному на каждой строке. Гарантируется, что при выполнении операции `pop` заданный вектор не будем пустым.

Формат выходных данных

Для кажой операции `push` выведите количество элементов в векторе.

Для кажой операции `pop` выведите значение удаляемого элемента.

Примеры

стандартный поток ввода	стандартный поток вывода
3 push 0 1 push 0 10 push 2 11 pop 0 push 0 23 pop 2	1 2 1 10 2 11

Задача 3. Сортировка строк

Имя входного файла:	`стандартный поток ввода`
Имя выходного файла:	`стандартный поток вывода`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

В данной задаче вам задано N пар строк. Из каждой входной пары строк получите две другие строки: конкатеницией первой строки со второй и второй с первой. Затем выведите получившиеся строки в отсортированном порядке.

Для решения этой задачи хорошо подойдут строки `std::string` из стандартной библиотеки. Использовать их достаточно легко:

```
#include <string>

// много кода

string left, right;
cin >> left >> right;
```

Не забывайте, что для строк определён удобный оператор сложения:

```
string s = left + right;
```

Совместите свои знанения о вектора и строках и решите данную задачу ;)

Формат входных данных

В превой строке входного файла задано целое число N.

Следующие N строк содержат по 2 строки, каждая из которых в длине не превосходит 20 символов и состоит только из строчных латинских букв.

Формат выходных данных

Выведите $2 \cdot N$ строк, отсортированных в лексикографическом порядке.

Примеры

стандартный поток ввода	стандартный поток вывода
3 a b vasya petya zoo keeper	ab ba keeperzoo petyavasya vasyapetya zookeeper

Замечания

Описание класса строки: `https://en.cppreference.com/w/cpp/string/basic_string`.

Задача 4. Геометрия врывается в этот простой пак задач

Имя входного файла:	`стандартный поток ввода`
Имя выходного файла:	`стандартный поток вывода`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Дано N точек на плоскости. Требуется отсортировать их по полярному углу.

Для решения этой в векторе можно хранить как и свою собственную структуру точки, так и использовать `std::pair` из стандартной библиотеки:

```
#include <utility>

// много кода

vector<pair<double, double>> v;
double x, y;
cin >> x >> y;
v.push_back({x, y});
```

Полярный угол можно вычислить с помощью встроенной в стандартную библиотеку функцию `atan2`: `https://en.cppreference.com/w/cpp/numeric/math/atan2`.

Формат входных данных

В превой строке входного файла задано целое число N.

Следующие N строк содержат по 2 вещественных числа. Гарантируется, что хотя бы одно из них не равно нулю.

Формат выходных данных

Выведите данные точки, отсортированные по полярному углу.

Примеры

стандартный поток ввода	стандартный поток вывода
3 2.3 3.1 1.2 -0.72 -3.4 0.0	2.3 3.1 -3.4 0.0 1.2 -0.72

Замечания

Описание структуры пары: `https://en.cppreference.com/w/cpp/utility/pair`.

Задача 5. M-M-Множество

Имя входного файла:	`стандартный поток ввода`
Имя выходного файла:	`стандартный поток вывода`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Требуется выполнить N операций над множеством целых чисел. Изначально множество пустое.

Операции бывают двух типов:

1. `+ x` — добавить число `x` в множество.
2. `? x` — проверить есть ли число `x` в множестве.

Для решения этой задачи удобно использовать контейнер `std::set` из стандартной библиотеки. Вам пригодятся методы `insert` для добавления элемента и `find` или `count` для поиска элемента во множестве.

Формат входных данных

В превой строке входного файла задано целое число N.

Следующие N строк содержат описанные опреации. Значение `x` по модулю не превосходит 10^{18} — используйте тип `int64_t` или `long long`.

Формат выходных данных

На каждую операцию проверки выведите "`da`", если число присутствует во множестве, и "`net`" в противном случае.

Примеры

стандартный поток ввода	стандартный поток вывода
10 ? 12 ? 20 + 12 + 15 ? 12 ? 20 + 12 + 20 ? 12 ? 20	net net da net da da

Замечания

Описание контейнера множества: `https://en.cppreference.com/w/cpp/container/set`.

Задача 6. M-M-Мульти-M-M-Множество

Имя входного файла:	`стандартный поток ввода`
Имя выходного файла:	`стандартный поток вывода`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Требуется выполнить N операций над множеством целых чисел. **Внимание**, в отличие от предыдущей задачи, во множестве допускается наличие одного значения несколько раз. Изначально множество пустое.

Операции бывают двух типов:

1. `+ x` — добавить число `x` в множество. Если `x` уже был во множестве, то добавится ещё один элемент `x` в него.
2. `- x` — удалить число `x` из множества. При этом, если во множестве было несколько вхождений `x`, то удаляются не все, а только одно. Если числа `x` во множестве не было, то данную операцию надо игнорировать.
3. `? x` — проверить есть ли число `x` в множестве.

Для решения этой задачи удобно использовать контейнер `std::multiset` из стандартной библиотеки.

Для удаления вам пригодится метод `erase`: `https://en.cppreference.com/w/cpp/container/multiset/erase` — обратите внимание, что при удалении по значению будут удалены все вхождения данного значения. Вам же для решения данной задачи больше подходит вариант удаления по итератору:

```
multiset<int64_t> m;
// много кода
m.erase(m.find(value));
```

Используйте `find` для поиска элемента во множестве.

Формат входных данных

В превой строке входного файла задано целое число N.

Следующие N строк содержат описанные опреации. Значение `x` по модулю не превосходит 10^{18} — используйте тип `int64_t` или `long long`.

Формат выходных данных

На каждую операцию проверки выведите "`da`", если число присутствует во множестве, и "`net`" в противном случае.

Примеры

стандартный поток ввода	стандартный поток вывода
9 ? 1 + 1 ? 1 + 1 ? 1 - 1 ? 1 - 1 ? 1	net da da da net

Замечания

Описание контейнера множества, допускающего наличие повторяющихся элементов: `https://en.cppreference.com/w/cpp/container/multiset`.

Задача 7. M-M-M-M-апа

Имя входного файла:	`стандартный поток ввода`
Имя выходного файла:	`стандартный поток вывода`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Вам дан лог отправки участинками решений в тестирующую систему. Требуется для каждой команды вывести id посылок, которые она совершила.

Формат входных данных

В превой строке входного файла задано цело число Z — количество записей в логе.

Следующие M строк содержат "лог" посылок. Каждая строка лога содержит название команды и id посылки, записанные через пробел. Название команды представляет из себя строку из строчных латинских букв, а id посылки — целое число не преводсходящее по модулю 10^{20}.

Гарантируется, что id посылок не повторяются. Хотя не то чтоб в этой задаче это было важно...

Формат выходных данных

Для каждой командые выведите информацию о её посылках в следующем формате: сначала идёт название команды, затем через пробел перечислены id посылок в том же порядке, в каком они были заданы во входном файле.

Каждую команду выводите на отдельной строке. Выводите команды в лексикографическом порядке.

Примеры

стандартный поток ввода	стандартный поток вывода
5 vasya 10 petya 20 misha 14 vasya 5 grisha 666	grisha 666 misha 14 petya 20 vasya 10 5

Замечания

Описание контейнера ассоциативного массива: `https://en.cppreference.com/w/cpp/container/map`.