Задача A. Быстрая сортировка

Имя входного файла:	`sort.in`
Имя выходного файла:	`sort.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Дан массив целых чисел. Ваша задача — отсортировать его в порядке неубывания с помощью алгоритма быстрой сортировки (qsort).

Формат входных данных

В первой строке входного файла содержится число N ($1 \leqslant N \leqslant 100\,000$) — количество элементов в массиве. Во второй строке находятся N целых чисел, по модулю не превосходящих 10^9.

Формат выходных данных

В выходной файл надо вывести этот же массив в порядке неубывания, между любыми двумя числами должен стоять ровно один пробел.

Примеры

sort.in	sort.out
10 1 8 2 1 4 7 3 2 3 6	1 1 2 2 3 3 4 6 7 8

Замечание

В этой задаче обязательно использовать быструю сортировку.

Задача B. Тестирующая система

Имя входного файла:	`ejudge.in`
Имя выходного файла:	`ejudge.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	64 мегабайта

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.

Примеры

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

Задача C. Anti-qsort test

Имя входного файла:	`anti-qsort.in`
Имя выходного файла:	`anti-qsort.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Рассмотрим алгоритм быстрой сортировки Хоара, с выбором в качестве барьерного элемента среднего элемента на отрезке (`q = A[(l + r) / 2]`):

```
void qsort(vector<int> & a, int left, int right)
// Сортировка A[left...right] включительно
{
    if (right <= left)
        return;
    int q = A[(l + r) / 2];
    int i = left;
    int j = right;
    while (i <= j) {
        while (a[i] < q)
            ++i;
        while (q < a[j])
            --j;
        if (i <= j) {
            swap(a[i], a[j]);
            ++i;
            --j;
        }
    }
```

```
    qsort(a, left, j);
    qsort(a, i, right);
}
```

По данному числу n составьте тест, являющийся перестановкой чисел от 1 до n, на котором этот алгоритм выполняет наибольшее число сравнений (подсчитываются сравнения `a[i] < q` и `q < a[j]`.

Формат входных данных

Программа получает на вход одно целое число n, $1 \leqslant n \leqslant 70\,000$.

Формат выходных данных

Программа должна вывести перестановку чисел от 1 до n, на котором данная реализация алгоритма быстрой сортировки Хоара будет выполнять наибольшее число сравнений.

Можно вывести любой из возможных ответов.

Примеры

anti-qsort.in	anti-qsort.out
3	1 3 2

Задача D. Рейтинг кавалеров

Имя входного файла:	`cavaliers.in`
Имя выходного файла:	`cavaliers.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Одна симпатичная и предприимчивая девушка ищет себе идеального партнера для танцевальной зарядки.

Идеальный кавалер по её представлениями должен иметь рост 180 см, поэтому прежде всего она хочет найти юношу, чей рост как можно ближе к 180 см. Будет ли кавалер выше или ниже указанной величины, не имеет значения (то есть юноши с ростом 179 и 181 для неё одинаково привлекательны).

Среди всех кандидатов одинаково подходящего роста ей нужен кто-то, чей вес насколько это возможно близок к 75 кг, но не превышает этой величины. Если же все кандидаты одного роста весят более 75 кг, то девушка выберет самого легкого из них.

Вам дан список кавалеров, содержащий имя партнера, его рост и вес.

Отсортируйте список по критерию привлекательности роста, при равной привлекательности роста — по привлекательности веса. При одинаковых параметрах роста и веса список должен быть отсортирован в лексикографическом порядке имен кавалеров.

Формат входных данных

В первой строке дано одно натуральное число N ($1 \leqslant N \leqslant 100\,000$). В последующих строках перечислены кавалеры по одному в строке в следующем формате: в начале идёт имя (строка из заглавных и строчных латинских букв, длиной не более 10), затем через пробел рост кавалера (натуральное число из диапазона $[1; 210]$), затем через пробел — вес (натуральное число из диапазона $[1; 120]$).

Формат выходных данных

Выведите отсортированный список имён кавалеров, по одному в строке.

Примеры

cavaliers.in	cavaliers.out
10 George 195 110 Thomas 180 75 John 180 75 James 180 65 Andrew 165 110 Martin 170 70 William 180 77 Franklin 195 70 Benjamin 165 70 Theodore 165 80	John Thomas James William Martin Benjamin Franklin Theodore Andrew George

Задача E. ACM Марафон

Имя входного файла:	`contest.in`
Имя выходного файла:	`contest.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Школьник Вася Иванов так сильно боялся идти на командную олимпиаду, что ему приснился кошмар: в ЛКШ вместо обычной олимпиады устраивали обязательный ACM-марафон. Это почти обычная командная олимпиада, отличается она только продолжительностью. Марафон длится ровно 24 часа, то есть если он начался в 00:00:00 то в 23:59:59 команда еще может сдать решение, а в 00:00:00 следующего дня — уже нет.

Как и в обычном турнире ACM, побеждает команда, решившая наибольшее число задач, а при равном количестве решенных задач лучше результат у той команды, у которой меньше штрафное время. Изначально штрафное время каждой команды равно нулю. За каждую правильно сданную задачу к штрафному времени команды прибавляют время в минутах, округленное вниз, прошедшее с начала соревнования до момента сдачи задачи. Кроме того, если зачтённой попытке предшествовало несколько неудачных попыток сдать ту же задачу, то за каждую из них к штрафному времени прибавляют двадцать минут. За неудачные попытки сдать задачу, которую команде в итоге так и не удалось решить, штрафного времени не начисляется. Так же посылки с результатом “Compilation error” и “Code style violation” не считаются неудачными, то есть за них не начисляются штрафные минуты.

Вам требуется написать программу, которая подсчитает результаты марафона.

Формат входных данных

В первой строке входного файла находится время начала олимпиады в формате

$hh : mm : ss$, где двухразрядное целое число hh ($0 \leqslant hh \leqslant 23$) означает час, а двухразрядные целые числа mm и ss ($0 \leqslant mm, ss \leqslant 59$) — минуты и секунды соответственно.

Во второй строке находится единственное целое число n ($1 \leqslant n \leqslant 1000$) — количество посылок за олимпиаду.

Далее следуют n строк с описаниями посылок. В начале каждой из них в двойных кавычках записано название команды, сделавшей посылку. Название может состоять из строчных и заглавных латинских букв, пробелов и цифр от 1 до 9. Длина названия — не меньше одного символа и не больше 255. После названия команды написано время посылки в том же формате, что и время начала контеста.

Далее через пробел идет заглавная латинская буква — номер задачи. Последние два символа в строке — результат посылки. Результат посылки может быть одни из следующих:
OK — OK
WA — Wrong answer
PE — Presentation error
TL — Time limit
ML — Memory limit
CE — Compillation error
CS — Code style violation

Примеры

contest.in	contest.out
00:00:00 5 "Super team" 00:00:23 A WA "Mega team" 00:10:21 A WA "Super team" 00:20:23 A OK "Mega team" 00:30:23 A OK "Mega team" 00:40:23 B OK	1 "Mega team" 2 90 2 "Super team" 1 40
01:00:00 3 "Team1" 01:10:00 A WA "Team1" 01:20:00 A OK "Team2" 01:40:00 B OK	1 "Team1" 1 40 1 "Team2" 1 40

Формат выходных данных

Выходной файл должен содержать итоговую таблицу результатов — по строке на каждую команду. Строки должны идти в порядке уменьшения результата, если у нескольких команд результаты равны, то порядок команд определяется названием — раньше идет та, название которой лексикографически меньше.

Каждая строка должна начинаться с места команды в итоговом зачете. Место команды — это $k + 1$, где k — число команд, имеющих строго лучший результат. Далее через пробел идет название команды в двойных кавычках, а за ним через пробел два числа — количество решенных задач и штрафное время.