Задача A. Проблема сапожника

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

В некоей воинской части есть сапожник. Рабочий день сапожника длится K минут. Заведующий складом оценивает работу сапожника по количеству починенной обуви, независимо от того, насколько сложный ремонт требовался в каждом случае. Дано n сапог, нуждающихся в починке. Определите, какое максимальное количество из них сапожник сможет починить за один рабочий день.

Формат входных данных

В первой строке вводятся числа K (натуральное, не превышает 1000) и n (натуральное, не превышает 500). Затем идет n чисел — количество минут, которые требуются, чтобы починить i-й сапог (времена — натуральные числа, не превосходят 100).

Формат выходных данных

Выведите одно число — максимальное количество сапог, которые можно починить за один рабочий день.

Примеры

cobbler.in	cobbler.out
10 3 6 2 8	2
3 2 10 20	0

Задача B. Создание архива

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

Системный администратор вспомнил, что давно не делал архива пользовательских файлов. Однако, объем диска, куда он может поместить архив, может быть меньше чем суммарный объем архивируемых файлов.

Известно, какой объем занимают файлы каждого пользователя.

Напишите программу, которая по заданной информации о пользователях и свободному объему на архивном диске определит максимальное число пользователей, чьи данные можно поместить в архив, при этом используя свободное место как можно более полно.

Формат входных данных

Программа получает на вход в одной строке число S — размер свободного места на диске ($0 \leqslant S \leqslant 10^7$) и число N — количество пользователей ($1 \leqslant N \leqslant 10^5$), после этого идет N натуральных чисел, не превосходящих 300 — объем данных каждого пользователя, записанных каждое в отдельной строке.

Формат выходных данных

Выведите наибольшее количество пользователей, чьи данные могут быть помещены в архив.

Примеры

archive.in	archive.out
100 2 200 50	1
100 3 50 30 50	2

Задача C. Выбор заявок

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

Вы прекрасно знаете, что в ЛКШ.Зима 2023 лекции читают лучшие преподаватели мира. К сожалению, лекционных аудиторий у нас не так уж и много, поэтому каждый преподаватель составил список лекций, которые он хочет прочитать ЛКШатам. Чтобы ЛКШата, утром идя на завтрак, увидели расписание лекций, необходимо его составить прямо сейчас. И без вас нам здесь не справиться.

У нас есть список заявок от преподавателей на лекции для одной из аудиторий. Каждая заявка представлена в виде временного интервала $[s_i, f_i)$ — время начала и конца лекции. Лекция считается открытым интервалом, то есть какая-то лекция может начаться в момент окончания другой, без перерыва. Необходимо выбрать из этих заявок такое подмножество, чтобы суммарно выполнить максимальное количество заявок. Учтите, что одновременно в лекционной аудитории, конечно же, может читаться лишь одна лекция.

Формат входных данных

В первой строке вводится натуральное число N, не более 1000 — общее количество заявок на лекции. Затем вводится N строк с описаниями заявок — по два числа в каждом s_i и f_i. Гарантируется, что $s_i < f_i$. Время начала и окончания лекции — натуральные числа, не превышают 1440 (в минутах с начала суток).

Формат выходных данных

Выведите одно число — максимальное количество заявок, которые можно выполнить.

Примеры

request.in	request.out
1 5 10	1
3 1 5 2 3 3 4	2

Задача D. Планирование заданий

Имя входного файла:	`schedule.in`
Имя выходного файла:	`schedule.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Имеется некоторое множество заданий и один исполнитель. На выполнение одного задания уходит единица времени. Задания можно выполнять начиная с момента времени 0. У каждого задания есть две характеристики: d_i и w_i. Если задание не было выполнено к моменту времени d_i, взимается штраф в размере w_i. Требуется минимизировать суммарный штраф.

Формат входных данных

Первая строка входного файла содержит натуральное число n — количество заданий ($1 \leqslant n \leqslant 1000$). Следующие n строк содержат по два натуральных числа, разделенных пробелом — d_i и w_i ($0 \leqslant d_i, w_i \leqslant 1\,000$).

Формат выходных данных

Выведите одно число — минимальный суммарный штраф.

Примеры

schedule.in	schedule.out
2 1 1 1 2	1
1 0 5	5

Задача E. Код Хаффмана

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

Заданы числа $p_1, p_2, \ldots, p_n$.

Предположив, что имеется текст, содержащий p_1 символов c_1, p_2 символов c_2, и т. д., постройте код Хаффмана и найдите суммарное число битов, необходимое для кодирования такого текста.

Формат входных данных

Первая строка входного файла содержит число n ($2 \leqslant n \leqslant 1000$). Вторая строка содержит n целых чисел $p_1, p_2, \ldots, p_n$ ($1 \leqslant p_i \leqslant 10^9$).

Формат выходных данных

Выведите одно число — число битов, необходимое для кодирования текста с заданным во входном файле количеством вхождений каждого символа.

Пример

huffman.in	huffman.out
10 1 2 3 4 5 6 7 8 9 10	173

Задача F. Алиса и яблоки

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

Алисе в стране чудес попались n волшебных яблок. Про каждое яблоко известно, что после того, как его съешь, твой рост сначала уменьшится на a_i сантиметров, а потом увеличится на b_i сантиметров. Алиса очень голодная и хочет съесть все n яблок, но боится, что в какой-то момент ее рост станет равным нулю или еще меньше, и она пропадет совсем. Помогите ей узнать, можно ли съесть яблоки в таком порядке, чтобы в любой момент времени рост Алисы был больше нуля.

Формат входных данных

В первой строке вводятся натуральные числа n и s ($1 \leqslant n \leqslant 1000$, $1 \leqslant s \leqslant 100\,000$) — число яблок и начальный рост Алисы. В следующих n строках вводятся пары натуральных чисел a_i, b_i, не больших 1000.

Формат выходных данных

Если яблоки съесть нельзя, выведите число -1. Иначе выведите n чисел — номера яблок, в том порядке, в котором их нужно есть.

Примеры

apples.in	apples.out
3 5 2 3 10 5 5 10	1 3 2
3 5 2 3 10 5 5 6	-1

Задача G. Минимальное покрытие

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

На прямой задано несколько отрезков с целочисленными координатами концов $[L_i, R_i]$. Выберите такое подмножество этих отрезков, что оно целиком покрывает отрезок $[0, M]$ и при этом состоит из минимально возможного количества элементов.

Формат входных данных

В первой строке находится целое число M ($1 \leqslant M \leqslant 5000$).

В следующих строках записана информация об отрезках — по одному на строку. Отрезок задаётся парой целых чисел L_i и R_i ($|L_i|, |R_i| \leqslant 50\,000$). Список завершается парой нулей. Количество отрезков не превышает 100 000.

Формат выходных данных

В первой строке выходного файла выведите минимальное количество отрезков, необходимое для покрытия отрезка $[0, M]$. Далее выведите координаты отрезков из покрывающего подмножества, упорядоченный по возрастанию координат левых концов отрезков.

Если покрыть отрезок $[0, M]$ нельзя, выведите единственную фразу «`No solution`».

Примеры

covering.in	covering.out
1 -1 0 -5 -3 2 5 0 0	No solution
1 -1 0 0 1 0 0	1 0 1