Задача A. ACM World Finals

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

Некоторые из вас, наверно, знают, что ежегодно проводится чемпионат мира по программированию среди студентов (http://acm.baylor.edu). В финал этого соревнования проходят около 80 команд со всего мира.

Каждая команда состоит из трех человек и имеет название. Напишите программу, которая по краткому названию команды и фамилиям ее участников, строит полное название команды.

Полное название команды состоит из краткого названия команды и списка фамилий ее участников. Фамилии участников в списке должны быть упорядочены по алфавиту и отделены друг от друга запятыми. Название команды от фамилий участников должно быть отделено двоеточием. После каждого знака препинания должен стоять ровно один пробел.

Формат входного файла

Входной файл содержит ровно 4 строки. Первая из строк содержит название команды. Каждая из следующих трех строк содержит фамилию одного из членов команды. Длины строк не превышают 50 символов.

Формат выходного файла

Выходной файл должен содержать ровно одну строку, содержащую полное название команды.

Примеры

acm.in	acm.out

`Dream Team`
`Knuth`
`Dijkstra`
`Cormen`
	`Dream Team: Cormen, Dijkstra, Knuth`

`Ivanovs Team`
`Ivanov`
`Ivanov`
`Ivanov`
	`Ivanovs Team: Ivanov, Ivanov, Ivanov`

`Team`
`a`
`aa`
`aaa`
	`Team: a, aa, aaa`

Задача B. Кубок CBOSS

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

В 2239 году команде-победителю Открытого кубка CBOSS (http://shade.msu.ru/~ejudge) достался весьма нетрадиционный приз — поездка по k самым красивым городам России. Так как в России красивых городов достаточно много, то победителям было предложено выбрать k городов из списка, содержащего n городов.

Для удобства занумеруем эти города целыми числами от 1 до n. Для каждого города известно t_i — время, требующееся на осмотр его достопримечательностей. Также для каждой пары $(i, j), 1 \leq i, j \leq n$ известно $a_{i,j}$ — время, которое требуется на проезд из i-ого города в j-ый. При этом может оказаться, что $a_{i,j} \neq a_{j,i}$, но $a_{i,i}$ всегда равно нулю.

У студентов, входящих в команду-победитель, не так много времени на посещение красивых городов, ведь скоро у них сессия. Поэтому они хотят выбрать k городов и посетить их в таком порядке, чтобы затраты времени были минимальны. Разумеется, посещать один город несколько раз им неинтересно. Также они не хотят приезжать в город, не осматривая при этом его достопримечательности.

Напишите программу, находящую нужные k городов и порядок, в котором их нужно посетить.

Формат входного файла

Первая строка входного файла содержит целые числа n и k ($1 \leq n \leq 7, 1 \leq k \leq n$). Каждая из последующих n строк входного файла содержит по n целых чисел каждая: j-ое число $(i+1)$-ой строки входного файла — это время, требуемое на проезд из i-ого города в j-ый ($a_{i,j}$). Последняя строка входного файла содержит n целых чисел $t_1, ..., t_n$.

Все числа во входном файле не превосходят 100. Все времена неотрицательны.

Формат выходного файла

В первой строке выходного файла выведите минимальное время, которое потребуется для посещения k городов с учетом осмотра достопримечательностей. Во второй строке выходного файла выведите номера городов в порядке посещения, гарантирующем такое время.

Примеры

cboss.in	cboss.out
4 3 0 3 2 1 8 0 6 5 1 2 0 4 5 6 7 0 1 2 3 4	10 3 1 4
4 4 0 3 2 1 8 0 6 5 1 2 0 4 5 6 7 0 1 2 3 4	18 3 1 4 2

Задача C. Интернет-олимпады

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

Каждый из вас знает, что Интернет-олимпиада, в которой вы участвуете, проводится в двух номинациях: базовой и усложненной. Ваша задача состоит в том, чтобы написать программу, обрабатывающую результаты Интернет-олимпиады, то есть определяющую, какие команды в какой номинации будут участвовать в следующей олимпиаде.

Правила перехода команд из одной номинации в другую таковы. Задачи усложненной номинации решают следующие команды:

- участвовавшие в предыдущей олимпиаде в усложненной номинации и решившие хотя бы одну задачу;
- участвовавшие в предыдущей олимпиаде в базовой номинации, решившие хотя бы одну задачу, и при этом решившие либо столько же задач, сколько команда-победитель, либо строго больше задач, чем команда, занявшая медианное место среди команд, решивших хотя бы одну задачу (место с номером k/2, где k — число команд, участвовавших в базовой номинации, решивших хотя бы одну задачу, округление производится вниз).

Все остальные команды участвуют в следующей олимпиаде в базовой номинации. В этой задаче мы считаем, что никакие новые команды не регистрируются для участия в следующей олимпиаде.

Напомним также, что при подведении итогов одной олимпиады команды сортируются по убывании количества решенных задач, а при равенстве количества решенных задач — по возрастанию штрафного времени.

Формат входного файла

Первая строка входного файла содержит два целых числа: n и m — соответственно, количество команд, участвовавших в базовой и усложненной номинациях ($1 \leq n, m \leq 1000$).

После этого идут n строк, описывающих результаты команд, участвовавших в базовой номинации. Каждая из этих строк содержит три целых числа, разделенных пробелами, — id, s, t — соответственно, уникальный идентификатор команды, количество решенных задач и штрафное время. Количество решенных задач — целое число от 0 до 12, а штрафное время — целое число от 0 до 20000. Идентификатор команды — это целое число от 1 до 2000.

После этого идут m строк, описывающих результаты усложненной номинации в таком же формате.

Формат выходного файла

В первой строке выходного файла выведите число k команд, которые допускаются до участия в усложненной номинации в следующей олимпиаде. Вторая строка должна содержать k целых чисел — идентификаторы этих команд в возрастающем порядке.

Примеры

<table>
<tr><th>io.in</th><th>io.out</th></tr>
<tr><td><pre>6 3
1 1 45
2 4 678
3 5 1000
4 5 894
5 2 343
6 3 555
1998 0 0
1999 1 34
2000 3 366</pre></td><td><pre>4
3 4 1999 2000</pre></td></tr>
</table>

Задача D. Java Challenge

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

Все участники олимпиад знают, что во время соревнования на счету каждая секунда. Иногда даже время, которое в суете затрачивается на переключение между окнами может оказаться критичным. В таких соревнованиях, как Java Challenge, количество окон может быть довольно большим (Java Challenge — это соревнование, проходящее в рамках финала чемпионата мира по программированию среди студентов. Оно состоит в разработке искусственного интеллекта для управления виртуальным роботом).

В данной задаче мы будем считать, что этот процесс выполняется следующим способом. В системе хранится циклический список открытых окон. При нажатии определенной комбинации клавиш k раз можно перейти в этом списке на k позиций в одну или в другую сторону. Кроме того, окна, относящиеся к каждому приложению так же организованы в циклический список. По этому списку также можно перемещаться в обе стороны, для перемещения на k позиций так же требуется k нажатий клавиш. При этом после своей активизации окно перемещаются в позицию перед первым элементом общего списка окон. Напишите программу, которая для каждого из окон будет определять минимальное количество нажатий клавиш, которое нужно затратить для его активизации. До начала выполнения операции активным является первое окно.

Формат входного файла

Первая строка входного файла содержит целое число n ($1 \le n \le 50000$) — количество открытых окон. Следующие n строк описывают окна в том порядке, в котором они идут в списке. Для каждого из окон задается номер приложения, которому соответствует это окно, и его номер в циклическом списке окон этого приложения.

Формат выходного файла

На единственной строке выхдоного файла для каждого окна выведите минимальное количество нажатий клавиш, которое надо затратить для его активации.

Примеры

java.in	java.out
9 3 2 2 2 2 3 1 3 2 1 3 1 4 1 1 2 1 1	0 1 2 2 2 1 3 2 1

Задача E. NEERC

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

В полуфинале студенческого чемпионата мира по програмиированию NEERC (http://neerc.ifmo.ru) участвуют команды из n институтов. Участники для проведения соревнований распределяются по k залам, каждый из которых имеет размеры, достачные для размещения всех команд от всех институтов. При этом по правилам соревнований в одном зале может находиться не более одной команды от института.

Многие институты уже подали заявки на участие в полуфинале. Оргкомитет полуфинала хочет допустить до участия максимально возможное количество команд. При этом, разумеется, должна существовать возможность рассадить их по залам без нарушения правил.

Напишите программу, определяющую максимальное количество команд, которые можно допустить до участия в полуфинале.

Формат входного файла

Первая строка входного файла содержит число n — число институтов, подавших заявки. Вторая строка входного файла содержит n чисел $a_1, \ldots, a_n$ (a_i — это количество команд, заявленных от института номер i). Последняя строка входного файла содержит число k — количество залов, в которых проходят соревнования.

Все числа во входном файле целые, положительные и не превосходят 10000.

Формат выходного файла

Выведите в выходной файл ответ на задачу.

Примеры

neerc.in	neerc.out
3 1 2 4 3	6
3 1 2 4 4	7

Задача F. Всероссийская олимпиада по информатике

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

Прошло уже много лет с тех пор, как состоялась первая Всероссийская олимпиада по информатике. Как и многие другие соревнования, наши олимпиады теперь проводятся в несколько дней. Теперь даже задача выбора подходящего времени для олимпиады представляет определенные трудности. Ведь на разных планетах, входящих в состав Российской Федерации используются разные способы отсчета времени: длина месяца, количество дней в неделе и те дни, по которым невозможно проведение олимпиады, могут различаться. Возникла необходимость написания программы, которая поможет решить эту задачу. И тогда в жюри вспомнят, что уже сейчас мы предвидели такую ситуацию и предожили вам решить подобную задачу.

В качестве первого шага найдите количество способов выбрать время проведения олимпиады.

Формат входного файла

В первой строке входного файла содержатся два целых числа n и l ($1 \le l \le n \le 100000$) — количество дней месяца и продолжительность олимпиады соответственно. Во второй строке задаются количество дней в неделе w, количество дней, запрещенных еженедельно, d_w и день недели, на который приходится первый день месяца s ($1 \le s \le w \le n$, $0 \le d_w \le w$). Треться строка содержит d_w номеров дней недели (например, выходных), в которые нельзя проводить олимпиаду. В четвертой строке записано количество дней месяца d_m, не подходящих для проведения олимпиады по причинам отличным от еженедельного распорядка (например, такими днями являются государственные праздники). Последняя строка содержит d_m целых чисел — номера этих дней. Дни месяца так же нумеруются начиная с 1. Заметим, что некоторые дни могут быть запрещенными сразу по обеим причинам.

Формат выходного файла

В выходной файл выведите единственное целое число — количество способов выбрать l подряд идущих дней, в которые возможно проведение олимпиады.

Примеры

roi.in	roi.out
31 3 7 1 7 7 2 1 9	15

Задача G. TopCoder

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

Некоторые из вас, наверное, слышали о сайте http://www.topcoder.com, на котором часто проводятся различные соревнования по программированию.

В некоторых из них участникам предлагаются три задачи, каждая из которых оценивается в некоторое количество баллов. В зависимости от того, насколько долго участник решал задачу, количество полученных им за нее баллов уменьшается. Как и в большинстве других соревнований, выигрывает участник, набравший наибольшее число баллов. Участники, набравшие одинаковое число баллов, считаются выступившими одинаково и их порядок в таблице итоговых результатов не важен. Из-за некоторых особенностей этих соревнований для предотвращения жульничества участники разделены в группы по 20 человек, называемые комнатами.

Ваша задача заключается в том, чтобы написать программу, которая по итоговым результатам в каждой комнате выводила бы суммарные итоговые результаты.

Формат входного файла

Первая строка входного файла содержит целое число n ($1 \le n \le 100$) — число комнат. Далее следуют n описаний итоговых результатов в комнатах.

Результаты в i-ой комнате заданы в следующем формате. Первая строка содержит целое число n_i ($1 \le n_i \le 20$)— количество участников в i-ой комнате. Следующие n_i строк содержат информацию о выступлениях участников. $j+1$-ая строка описания результатов в i-ой комнате содержит информацию об участнике, занявшем в i-ой комнате j-ое место: разделенные одним пробелом вещественное число $total_j^i$ ($-5000 \le total_j^i \le 10000$) и строку $name_j^i$ — соответственно количество набранных участником баллов и его имя. Имя участника имеет длину от 1 до 25 и может содержать только буквы латинского алфавита, цифры и символ подчеркивания. При этом первый символ имени не является цифрой. Все вещественные числа заданы с двумя знаками после десятичной точки.

Гарантируется, что в каждой комнате участники упорядочены по невозрастанию набранных ими баллов.

Формат выходного файла

На первой строке выходного файла выведите N — суммарное число участников. На следующих N строках выведите информацию о выступлении участников. $k+1$-ая строка описания суммарных результатов должна содержать информацию об участнике, занявшем k-ое место: разделенные одним пробелом вещественное число $total_k$ с двумя знаками после десятичной точки и строку $name_k$ — соответственно количество набранных участником баллов и его имя.

Не забудьте, что участники должны быть упорядочены по невозрастанию набранных ими баллов.

Примеры

topcoder.in	topcoder.out
2 6 909.94 Savior 439.51 tywok 130.52 LimberG 0.00 BryanChen 0.00 angsa -75.00 The_Hedgehog 5 867.15 Ying 448.12 natori 195.32 aubergineanode 0.00 shalinmangar -25.00 Excilus	11 909.94 Savior 867.15 Ying 448.12 natori 439.51 tywok 195.32 aubergineanode 130.52 LimberG 0.00 angsa 0.00 shalinmangar 0.00 BryanChen -25.00 Excilus -75.00 The_Hedgehog

Задача H. Test-The-Best

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

Конкурс Test-the-best (http://www.test-the-best.by/), в котором участвуют лучшие программисты из Беларуси, России и других стран, проводит очный тур. Стараясь не отставать от времени, организаторы решили позаботиться о безопасности. В последнее время на рынке техники есть много аппаратуры, позволяющей осуществлять видеонаблюдение.

Широко распространены программы, позволяющие автоматически обрабатывать полученные результаты. Однако имея дело с участниками этих соревнований, на существующие разработки полагаться небезопасно. Поэтому у оргкомитета возникла необходимость написания собственной системы анализа изображений. Перед вами поставлена задача сравнения двух черно-белых изображений на клетчатой сетке. Изображения считаются одинаковыми, если множества черных пикселей в них могут быть получены друг из друга поворотом на 90, 180, или 360 градусов и, возможно, симметрией относительно вертикальной оси.

Формат входного файла

Входной файл содержит описания двух изображений в следующем формате: первая строка содержит два целых числа n и m ($1 \le n, m \le 500$) — высоту и ширину изображения соответственно. Затем следуют n строк, содержащих по m символов: '`#`' обозначает черный пиксель, '`.`' — белый.

Формат выходного файла

В выходной файл выведите одно слово: «`Yes`», если изображения одинаковы и «`No`» в противном случае.

Примеры

<table>
<tr><th>ttb.in</th><th>ttb.out</th></tr>
<tr><td><pre>7 8
....
..###...
..#.....
....
....
....
....
6 10
....
...#......
...#......
...##.....
....
..........</pre></td><td><pre>Yes</pre></td></tr>
<tr><td><pre>1 1
#
1 1
.</pre></td><td><pre>No</pre></td></tr>
</table>