Для всех задач:

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	`1 секунда на тест`
Ограничение по памяти:	`64 Мб`

Задача 5. «Спички» (Транзитивное замыкание графа)

Во время игры какое-то количество спичек рассыпается по столу и игроки должны убрать их, выбирая по одной, не сдвинув с места остальные спички. Вам нужно определить, касаются ли данные спички друг друга. Вам будет дан список координат концов спичек, и нужно будет определить, соединены ли две данные спички или нет. Заметим, что касание — это соединение, и что две спички могут быть соединены, если они соединены не прямо, а посредством других спичек.

Входные данные

Входной файл сдержит несколько тестов. В первой строке каждого теста находится целое число n ($1 < n < 13$), задающее количество спичек на столе. Каждая из следующих n строк содержит 4 положительных целых x_1, y_1, x_2, y_2, обозначающих координаты (x_1, y_1), (x_2, y_2) концов одной спички. Все координаты меньше 100. (Заметим, что спички могут быть разной длины). Первая введенная спичка будет спичкой номер 1, вторая, соответственно, номер 2 и т.д. остальные строки теста содержат по два целых числа a и b (от 1 до n, включительно). Вам нужно определить, связана ли спичка a со спичкой b, или нет. Конец теста : a = b = 0. Все данные корректны и нет спичек нулевой длины.
После каждого теста во входном файле пустая строка.
Последняя строка в файле содержит один символ — 0.

Выходные данные

Вам нужно для каждой пары спичек сгенерировать строчку **`CONNECTED`**, если данные спички связаны, и **`NOT CONNECTED`**, наоборот. Будем считать, что спичка связана сама с собой.
Пустые строки между тестами не вставлять.

Пример

input.txt	*output.txt*
7 1 6 3 3 4 6 4 9 4 5 6 7 1 4 3 5 3 5 5 5 5 2 6 3 5 4 7 2 1 4 1 6 3 3 6 7 2 3 1 3 0 0 0	CONNECTED NOT CONNECTED CONNECTED CONNECTED NOT CONNECTED CONNECTED

Задача 6. «Наложение рамок» (Топологическая сортировка)

Рассмотрим следующие рамки, расположенные в массивах размерностью 9 x 8.

```
........  ........  ........  ........  .CCC....
EEEEEE..  ........  ........  ..BBBB..  .C.C....
E....E..  DDDDDD..  ........  ..B..B..  .C.C....
E....E..  D....D..  ........  ..B..B..  .CCC....
E....E..  D....D..  ....AAAA  ..B..B..  ........
E....E..  D....D..  ....A..A  ..BBBB..  ........
E....E..  DDDDDD..  ....A..A  ........  ........
E....E..  ........  ....AAAA  ........  ........
EEEEEE..  ........  ........  ........  ........
  1         2         3         4         5
```

Теперь наложим их одна на другую, начиная с первой внизу и заканчивая пятой наверху. Если часть рамки закрывает другую, то она скрывает нижнюю часть. Полученная стопка рамок будет выглядеть следующим образом:

```
.CCC....
ECBCBB..
DCBCDB..
DCCC.B..
D.B.ABAA
D.BBBB.A
DDDDAD.A
E...AAAA
EEEEEE..
```

В каком порядке рамки складывались, начиная снизу? Ответом будет EDABC.
Ваша задача определить порядок, в каком рамки накладывались друг на друга, начиная снизу по заданной картинке результирующей стопки рамок.
Правила следующие: Ширина рамки – это одна литера, а стороны рамки не могут быть короче 3 литер. По крайней мере, часть каждой из четырех сторон рамки можно увидеть. Угол показывает две стороны. Рамки обозначаются заглавными латинскими буквами. Разные рамки обозначаются разными буквами.

Входные данные

Каждый блок входного файла содержит высоту h ($h \le 30$) на первой строке и ширину w ($w \le 30$) на второй. Далее дана картинка сложенных рамок в виде h строк w литер в каждой.
Ваш входной файл может содержать несколько тестов, заданных в описанном выше формате, без пустых строк между тестами. Все тесты из входного файла должны обрабатываться последовательно.

Выходные данные

В выходной файл в качестве решения записывается строка, состоящая из букв рамок в той последовательности, в которой они складывались, начиная снизу. Если существует несколько вариантов их расположения, нужно перечислить все варианты в алфавитном порядке, каждый на отдельной строке. Всегда существует, по крайней мере, один вариант расположения рамок для каждого теста. Пустых строк не вставлять.

Пример

input.txt	*output.txt*
9 8 .CCC.... ECBCBB.. DCBCDB.. DCCC.B.. D.B.ABAA D.BBBB.A DDDDAD.A E...AAAA EEEEEE..	EDABC

Задача 7. Шлю я за пакетом пакет... (Алгоритм Дейкстры)

В базе данных роутера хранится информация о нескольких серверах, некоторые из них связаны между собой напрямую, другие – только опосредованно. Получая электронное письмо от сервера с номером M (отправителя), предназначенное для сервера с номером N (получателя), роутер должен найти в своей базе данных самый короткий путь пересылки этого письма по сети. Напишите программу, которая, зная информацию о всевозможных соединениях и их длительности, осуществляла бы поиск самого короткого пути пересылки и выдавала бы информацию о времени, за которое письмо преодолеет весь этот путь. Если такого пути нет (например, на промежуточном сервере произошла авария), то необходимо выдать сообщение '**no**'.

Входные данные

В первой строке входного файла *input.txt* содержится одно натуральное число N – количество серверов, информация о которых записана в базе данных роутера($1 \le N \le 100$).

Во второй строке – два целых числа $1 \le S_1, S_2 \le N$, разделенные пробелом – номера сервера-отправителя и сервера-получателя.

Начиная с третьей строки и до конца файла, записаны имеющиеся между серверами активные каналы связи и скорость передачи данных по этим каналам. Сначала записаны номера серверов $1 \le S_i, S_j \le N$, а затем скорость передачи данных между ними – целое число $0 \le K \le 1000$. Все числа в одной строке разделены пробелами.

Выходные данные

В выходной файл *output.txt* нужно записать одно целое число – время, необходимое письму для прохождения по самому быстрому пути связи.

Пример

input.txt	*output.txt*
4 4 1 1 2 10 1 3 2 2 4 1 3 4 10	11
4 1 4 1 2 100 1 3 20 2 3 57	no

Задача 8. "Дороги" (Минимальный каркас графа)

Остров Флатландия идеально плоский. К сожалению, на Флатландии крайне плохая система дорог. Правительство Флатландии, беспокоясь о данной проблеме, построило несколько дорог, соединяющих некоторые из самых важных городов. Но, тем не менее, осталось несколько городов, до которых все еще нельзя добраться по дорогам.
Нужно построить больше дорог, чтобы можно было проехать между любыми двумя городами, не съезжая с дороги.
Города Флатландии имеют номера от 1 до *N,* i-ый город находится в точке, с координатами (x_i, y_i). Каждая дорога соединяет два города. Все дороги (как уже построенные, так и нуждающиеся в постройке) представляют собой прямые линии, длина которых равняется расстоянию в координатах между городами. По всем дорогам можно проехать в обе стороны. Дороги могут пересекаться, но водитель может сменить дорогу только в каком-либо городе.
Правительство Флатландии хочет мимимизировать стоимость постройки новых дорог, но, вместе с тем, оно хочет гарантировать достижимость любого города из любого другого. Так как Флатландия плоская, стоимость дороги всегда пропорциональна ее длине. То есть, самая дешевая система дорог – та, в которой минимальна сумма длин дорог.

Входные данные
Входной файл состоит из двух частей. Первая часть описывает все города страны, а вторая – все уже построенные дороги.
В первой строке файла *input.txt* содержится одно натуральное число N ($1 \le N \le 750$) – количество городов. Следующие N строк содержат по два целых числа x_i и y_i, разделенных пробелом - координаты i-го города ($1 \le i \le N$). Координаты по модулю не превосходят 10000. Координаты никаких двух городов не совпадают.
В следующей строке содержится одно натуральное число M ($0 \le M \le 1000$) – количество построенных дорог.
Следующие M строк содержат по два целых числа, разделенных пробелом. Эти два числа задают пару городов, соединенных дорогой. Каждые два города соединены максимум одной дорогой.

Выходные данные
В выходной файл нужно вывести список подлежащих постройке дорог с минимальной суммой длин. Каждая дорога описывается двумя числами, разделенными пробелом, – номерами городов, соединенных данной дорогой.
Если дороги строить не надо (все города уже соединены), тогда выходной файл надо оставить пустым.

Пример

input.txt	*output.txt*
9 1 5 0 0 3 2 4 5 5 1 0 4 5 2 1 2 5 3 3 1 3 9 7 1 2	1 6 3 7 4 9 5 7 8 3

Задача 10a "Делимость"

Рассмотрим случайную последовательность целых чисел. Мы можем расставить знаки + и – между числами в последовательности, получая при этом различные арифметические выражения, принимающие различные значения. Возьмем, к примеру, последовательность 17, 5, –21, 15. Из нее можно получить восемь различных выражений:
17 + 5 + -21 + 15 = 16
17 + 5 + –21 – 15 = –14
17 + 5 – –21 + 15 = 58
17 + 5 – –21 – 15 = 28
17 – 5 + –21 + 15 = 6
17 – 5 + –21 – 15 = –24
17 – 5 – –21 + 15 = 48
17 – 5 – –21 – 15 = 18
Назовем последовательность **делимой на K**, если можно расставить знаки + и – между числами из последовательности так, что результат получившегося выражения будет делиться на K. В примере выше последовательность делится на 7 (17+5+–21–15=–14), но не делится на 5.
Ваша задача – написать программу, определяющую делимость данной последовательности.
Входные данные
Первая строка входного файла содержит два целых числа N и K ($1 \le N \le 10000$, $2 \le K \le 100$), разделенных проеблом.
Вторая строка содержит последовательность из N целых чисел, разделенных пробелами. Числа не превосходят 10000 по модулю.

Выходные данные

Выведите в выходной файл слово "**`Divisible`**", если последовательность делится на K или "**`Not divisible`**" в противном случае.

Пример

input.txt	*output.txt*
`4 7` `17 5 -21 15`	`Divisible`
`4 5` `17 5 -21 15`	`Not divisible`

Задача 10б "Гангстеры"

N гангстеров собрались в ресторан. i-й гангстер приходит в момент времени T_i и имеет *ценность* P_i. Дверь ресторана имеет K+1 *степеней открытости,* принимающих целочисленные значения из диапазона [0,K]. Степень открытости может измениться только на единицу за раз: увеличиться на единицу, уменьшиться или не измениться вообще. В начальный момент времени дверь закрыта (степень открытости 0). i-й гангстер входит в ресторан только если степень открытости двери совпадает с его *толщиной* S_i. Если в момент прихода гангстера степень открытости двери не равна его толщине, то гангстер обижается, уходит и никогда уже не возвращается.
Время работы ресторана – интервал [0,T].
Ваша цель – открывая и закрывая дверь, собрать гангстеров в ресторане так, чтобы их суммарная ценность была максимальной.

Входные данные
Первая строка входного файла содержит числа N, K и T, разделенные пробелами. ($1 \le N \le 100$, $1 \le K \le 100$, $0 \le T \le 30000$)
Вторая строка содержит моменты прихода гангстеров в ресторан $T_1, T_2, ..., T_N$, разделенные пробелами. ($0 \le T_i \le T$ i = 1, 2, ..., N).
В третьей строке вы найдете ценности гангстеров $P_1, P_2, ..., P_N$, разделенные пробелами. ($0 \le P_i \le 300$, i = 1, 2, ..., N).
Толщины же гангстеров $S_1, S_2, ..., S_N$, разделенные пробелами, ищите в четвертой строке. ($1 \le S_i \le K$, i = 1, 2, ..., N).
Все значения, по традиции, целочисленные.

Выходные данные
Выведите в выходной файл одно целое число – максимальную ценность гангстеров в ресторане. Если ни один из них так и не сможет войти в ресторан, выведите 0.

Пример

input.txt	*output.txt*
4 10 20 10 16 8 16 10 11 15 1 10 7 1 8	26
2 17 100 5 0 50 33 6 1	0