Задача 1. Подземелье

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Ходислав готовит приключение по настольной ролевой игре «Подземелья и драконы», подземелье он уже спроектировал, теперь хочет понять, поместится ли в него дракон.

Подземелье представляет собой прямоугольное клетчатое поле размером $R \times C$, свободные клетки которого закодированы символом '.', а горная порода — символом '#'.

Верхняя левая клетка находится в нулевой строке и нулевом столбце, строки нумеруются сверху вниз, а столбцы — слева направо. Пустые клетки образуют односвязную область без горной породы внутри. На границе подземелья всегда находится горная порода, т.е. строки 0 и $R-1$, а также столбцы 0 и $C-1$ составлены только из '#'.

Чтобы понять, получится ли разместить дракона в подземелье, Ходиславу необходимо получить список клеток, при соединении левых верхних углов которых получается контур подземелья, то есть простой многоугольник, со сторонами, параллельными осям координат, который отделяет пустые клетки от горной породы.

Требуется вывести строку и столбец соответствующих клеток в порядке обхода против часовой стрелки, начиная с самой левой клетки (если таких клеток несколько, то с самой верхней из них); количество клеток в списке должно быть минимальным.

Формат входных данных

В первой строке записаны целые числа R и C ($3 \leqslant R, C \leqslant 1000$) — размеры поля. Следующие R строк длины C состоят из символов '.' и '#' и задают подземелье. Гарантируется, что в подземелье есть хотя бы одна пустая клетка и что множество пустых клеток удовлетворяет условиям задачи.

Формат выходных данных

В первой строке выведите единственное целое число — количество клеток в списке, далее в отдельных строках выведите через пробел пары «строка-столбец» для клеток из списка в порядке обхода контура против часовой стрелки, начиная с самой верхней среди самых левых клеток.

Замечание

Обратите внимание, что выводить надо пары «строка-столбец», а не декартовы координаты (в частности, позиция по вертиикали выводится **первой**, а не второй).

Пример

стандартный ввод	стандартный вывод
3 3 ### #.# ###	4 1 1 2 1 2 2 1 2
4 5 ##### ##.## #...# #####	8 2 1 3 1 3 4 2 4 2 3 1 3 1 2 2 2

Пояснение к примеру

Разберём второй пример.

```
  01234
0 #####
1 ##GF#
2 #AHED
3 #B##C
```

Левые верхние углы клеток ABCDEFGH при соединении в указанном порядке дают требуемый контур. Так как контур является восьмиугольником, то меньшим числом клеток обойтись нельзя.

Задача 2. Викторина

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Пациенты черноморской психбольницы придумали новую игру. Один, обычно это наиболее продвинутый из них, Берлага, отвечает на вопросы радиовикторины, за что ему и начисляются очки по следующему правилу: за правильные ответы он получает очки в плюс, за неправильные — в минус. При этом за каждый последующий правильный (или неправильный) ответ абсолютное значение прибавляемых (или вычитаемых – для неправильного ответа) баллов увеличивается на 1, если предыдущий тоже был правильным (или неправильным – для неправильного).

Если же ответ поменял свою истинность, то отсчет очков в плюс или в минус начинается с единицы.

Например, если была дана следующая цепочка ответов $-+++++---+$ («+» — правильный ответ, «−» — неправильный), то Берлага получит за свои ответы на эти девять вопросов соответственно -1, $+1$, $+2$, $+3$, $+4$, -1, -2, -3, $+1$ очко, что даёт в сумме четыре очка.

Вам предлагается написать программу, которая по значению набранной Берлагой суммы и количеству данных им правильных и неправильных ответов определяет, могла ли случиться такая игра, и в случае, если могла, строит пример такой игры.

Формат входных данных

В первой строке входного файла записано целое число Q — количество тестовых примеров ($1 \leqslant Q \leqslant 1000$).

Каждая из последующих Q строк задаёт один тестовый пример. В каждой строке записано по три целых числа S, T и F — общее количество набранных очков, количество правильных и неправильных ответов, соответственно ($-10^6 \leqslant S \leqslant 10^6, 0 \leqslant T \leqslant 100, 0 \leqslant F \leqslant 100$).

Формат выходных данных

Для каждого тестового примера в выходной файл необходимо вывести вердикт — можно или нельзя набрать указанное количество очков при заданных величинах правильных и неправильных ответов.

Если это возможно, то нужно вывести две строки: в первой написать слово `YES`, а во второй — последовательность ответов, которые дают нужную сумму. Если таких последовательностей существует несколько, разрешается вывести любую из них.

Если набрать указанное количество очков невозможно, то выдать одну строку со словом `NO`.

Пример

стандартный ввод	стандартный вывод
2 6 4 1 3 4 1	YES +++-+ NO

Задача 3. Меряем крутизну

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Учебный самолёт оборудован двумя приборами. Прибор для измерения высоты представляет набранную высоту в виде доли от максимально допустимой высоты полёта — числа от 0 до 1 в виде периодической дроби с периодом h фиксированной длины q и сохраняет в памяти n_h полных периодов. Например, если $h = 314$, $q = 3$ и $n_h = 2$, то будет сохранена дробь 0.314314.

Прибор для измерения длины представляет длину в виде доли от максимально допустимой дальности полёта — числа от 0 до 1 в виде периодической дроби с периодом ℓ фиксированной длины r и сохраняет в памяти n_ℓ полных периодов. Например, если $\ell = 15$, $r = 2$ и $n_\ell = 4$, то будет сохранена дробь 0.15151515.

Крутизна взлёта вычисляется как угол AOB, где O – точка отрыва от земли, A — точка, в которой самолёт перешёл к горизонтальному полёту, B — проекция этой точки на поверхность земли (в данной задаче рельефом и кривизной поверхности Земли можно пренебречь). При вычислении крутизны используются полученные от приборов и сохранённые в памяти значения (то есть 0.341341 и 0.15151515 в случае разобранных выше примеров параметров).

Есть данные о двух учебных полётах. В первом полёте $h = h_1$, $\ell = \ell_1$, во втором — $h = h_2$, $\ell = \ell_2$. Выясните, для какого из двух полётов вычисленная крутизна будет больше.

Формат входных данных

Первая строка входных данных содержит шесть целых чисел h_1, h_2, ℓ_1, ℓ_2, n_h и n_ℓ соответственно ($1 \leqslant h_1, h_2, \ell_1, \ell_2, n_h, n_\ell \leqslant 10^9$) — значение периода для высоты первого и второго полёта, значение периода для длины первого и второго полёта и количество сохраняемых в памяти периодов для высоты и для длины соответственно.

Гарантируется, что существует целое q такое, что $10^{q-1} \leqslant h_1, h_2 < 10^q$ и целое r такое, что $10^{r-1} \leqslant \ell_l, \ell_2 < 10^r$.

Формат выходных данных

Выведите 1, если вычисленная крутизна взлёта первого полёта больше вычисленной крутизны взлёта второго полёта, -1, если вычисленная крутизна взлёта второго полёта больше вычисленной крутизны взлёта первого полёта, и 0, если значения равны.

Пример

input.txt	output.txt
10 20 300 399 1 1	-1

Задача 4. Погрузка

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мебибайт

По этой задаче допускается неточное решение, задача влияет на рейтинг особым образом.

Днём Ходислав водит приключения по настольной ролевой игре «Подземелья и драконы», а ночью грузит тары в фуру.

Фура — прямоугольник шириной 2 400 мм и длиной 60 000 мм (Ходислав уверен, что гигантская 60-метровая фура действительно существует). Тары — тоже прямоугольники. На складе, где работает Ходислав, используются тары трёх размеров: 1200×800, 900×600 и 850×735 мм.

Ходиславу нужно погрузить в фуру n единиц тары размера 1200×800 мм, k единиц тары размера 900×600 мм и l единиц тары размера 850×735 мм.

Стороны тары должны быть параллельны сторонам фуры, допускается поворачивать тару на 90°. Тара может касаться (по углу или стороне) другой тары или внутренней части границы фуры, пересечения запрещены.

Определим *длину укладки*, как расстояние вдоль длины фуры от левого края фуры до самой правой стороны уложенных тар. Иными словами, это максимальная координата y в вашем ответе (см. «Формат выходных данных»).

Чем меньше длина укладки, тем больше балл (см. описание системы оценки).

Формат входных данных

В первой и единственной строке записаны целые неотрицательные числа n, k и l ($1 \leqslant n + k + l \leqslant 50$) — количество тары соответствующего размера, которое нужно уложить.

Формат выходных данных

Необходимо вывести три строки, содержащих описание укладки n, k и l тар соответствующего размера. Если n, k или l равно 0, оставьте соответствующую строку пустой.

Описание укладки, содержащее m однотипных тар, состоит из $4m$ неотрицательных целых чисел, каждая тара описывается четырьмя числами — координатами левого верхнего и правого нижнего углов, записанными через пробел: $x_0 \_ y_0 \_ x_1 \_ y_1$. Внутри одной строки координаты тар отделяются пробелом.

Ось x направлена сверху вниз вдоль ширины фуры, ось y направлена слева направо вдоль длины фуры. Левый верхний угол фуры имеет координату $(x, y) = (0, 0)$, единица измерения — мм. Любая координата (x, y) в ответе должна удовлетворять следующим ограничениям: $0 \leqslant x \leqslant 2400, 0 \leqslant y \leqslant 60000$.

Примеры

стандартный ввод
2 0 0
стандартный вывод
0 0 1200 800 1200 0 2400 800

стандартный ввод
3 7 5
стандартный вывод
0 0 1200 800 0 2485 1200 3285 1600 3300 2400 4500 1200 0 1800 900 1800 0 2400 900 1800 900 2400 1800 1800 1800 2400 2700 1200 2700 2100 3300 0 3285 600 4185 600 3300 1500 3900 0 800 850 1535 850 900 1700 1635 0 1535 735 2385 735 1635 1470 2485 600 3900 1450 4635

Система оценки

Баллы за тест определяются, как отношение длины укладки для решения жюри и для вашего решения. Таким образом, если решение выдаёт в три раза более длинную укладку, чем решение жюри, то оно получает одну треть балла за тест. За тесты с результатом проверки не ACCEPTED вы получаете 0 баллов.

Баллы за всю задачу определяются, как сумма баллов решения по всем тестам, всего тестов — 40 штук. В задаче есть четыре уровня сложности, и чтобы решить каждый уровень, нужно набрать суммарные баллы больше заданного барьера:

1. 38
2. 39.1
3. 39.8
4. 40

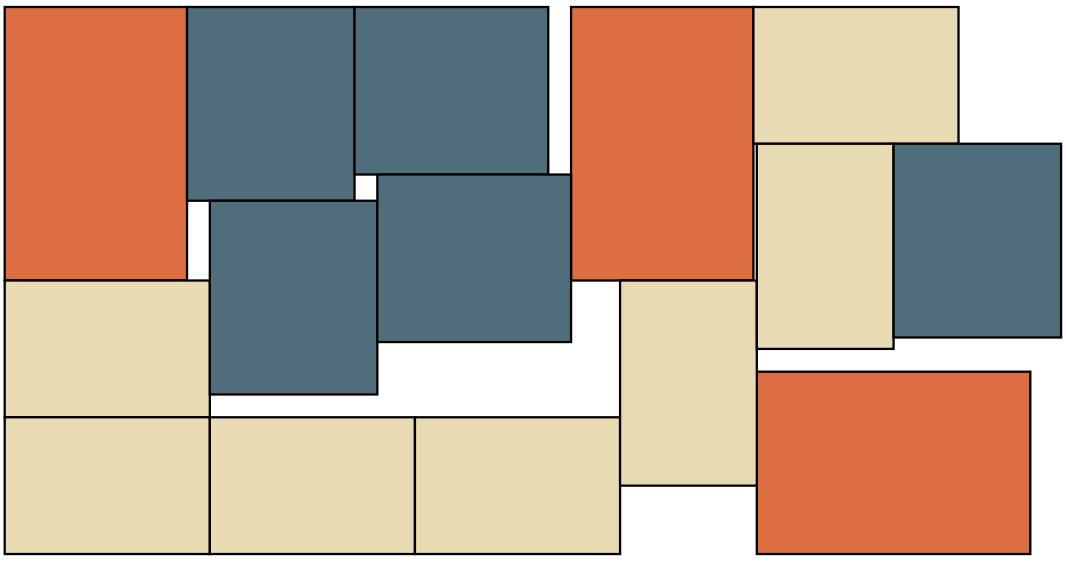
За преодоление каждого барьера вы получаете $+\frac{1}{4}$ задачи в рейтинге. Например, если вы прошли второй барьер и, кроме этой задачи, решили ещё 5 «обычных» задач, то количество задач в рейтинге будет равно 5.5.

Время сдачи этой задачи определяется по максимальному решённому уровню — по первой посылке, которая преодолела соответствующий барьер. Штрафное время начисляется за посылки, которые были до посылки, по которой определялось время сдачи. Во время тура частота посылок по этой задаче может быть ограничена: например, не чаще раза в минуту.

Пояснение к примеру

Во втором тесте из условия на самом деле три строки: первая, вторая вместе с третьей (разрезаны из-за форматирования) и четвёртая.

Ниже приведена визуализация укладки для второго теста из условия. Ваша укладка может отличаться от приведённой.



Задача 5. Генетика

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	3 секунды
	4 секунды для java и kotlin
Ограничение по памяти:	256 мебибайт

В институте цитологии и генетики умеют копировать, разрезать, склеивать и разворачивать цепочки нуклеотидов. Цепочка нуклеотидов представляет собой строку, состоящую из букв «a», «c», «g» и «t».

Учёные собираются проделать некоторое количество действий над такими цепочками и найти среди них пары совпадающих. Чтобы облегчить труд учёных, просимулируйте ту работу, которую они собираются проделать.

Формат входных данных

Первая строка входного файла содержит единственное число Q — количество действий учёных института ($1 \leqslant Q \leqslant 5 \cdot 10^4$). Далее следуют Q строк, в каждой из которых задано одно из следующих действий:

- `ADD` s — добавляет цепочку, представленную данной непустой строкой s, в набор рассматриваемых цепочек, ей присваивается очередной номер по порядку;
- `DEL` n — удаляет из набора цепочку с заданным номером n (её номер не освобождается и далее его использовать нельзя);
- `JOIN` n_1 n_2 — создаёт новую цепочку, представляющую собой конкатенацию двух имеющихся в наборе цепочек с номерами n_1 и n_2, ей присваивается очередной номер, цепочки n_1 и n_2 не удаляются.
- `CUT` n p — создаёт две новых цепочки, представляющих собой префикс и суффикс в позиции p имеющейся в наборе цепочки с номером n, им присваиваются номера по порядку (сначала префиксу, затем суффиксу); p может принимать значения от нуля до длины цепочки с номером n включительно (обратите внимание, что в случаях $p = 0$ или p, равного длине цепочки с номером n, в набор добавляется пустая цепочка); цепочка n при этом также остаётся в наборе.
- `REV` n — создаёт новую цепочку, представляющую собой переписанную в обратном порядке (справа налево) цепочку с номером n, ей присваивается очередной номер.

Нумерация цепочек начинается с нуля. Изначально набор цепочек пуст. Гарантируется, что все цепочки, с которыми работают учёные, содержат не более 10^5 нуклеотидов. Гарантируется, что суммарная длина цепочек, добавленных действием `ADD` не превышает $2 \cdot 10^5$.

Формат выходных данных

Каждый раз, когда в результате какого-либо действия появляется цепочка, совпадающая с уже имеющейся в наборе (пока не удалённой), выведите строку вида «`[`i`] == [`j`]`», где i — номер, который получила новая цепочка, а j — номер совпадающей с ней уже имеющейся в наборе цепочки. В случае, если есть несколько подходящих j, нужно выводить максимальный.

Пример

стандартный ввод	стандартный вывод
10 ADD ag ADD tc ADD agtc JOIN 0 1 ADD ctga REV 3 CUT 3 2 CUT 5 2 REV 6 REV 7 JOIN 1 7 JOIN 7 1 DEL 13 JOIN 7 7	[3] == [2] [5] == [4] [6] == [0] [7] == [1] [10] == [9] [11] == [8] [13] == [12] [14] == [12]

Задача 6. Логистика

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мебибайт

В городе Н. N магазинов сети «FTP» и M льготных маршрутов такси, соединяющих некоторые из магазинов. Маршруты являются односторонними (то есть наличие льготной ставки на маршрут из A в B не гарантирует льготной ставки на маршрут из B в A). Каждый магазин имеет свой тип.

Экспедитор стартует из магазина при главном складе (магазин имеет уникальный тип «^») и завершает работу в центральном магазине сети в здании штаб-квартиры «FTP» (магазин имеет уникальный тип «$»). При этом экспедитор должен посетить некую последовательность магазинов, обладающих следующим свойством: последовательность типов этих магазинов содержит некоторую заданную последовательность типов S как подпоследовательность.

При этом согласно правилам внутреннего распорядка компании экспедитор может перемещаться только на такси и только по льготным маршрутам.

Ваша задача — определить минимальную сумму, которую экспедитор потратит на такси при соблюдении этих условий.

Формат входных данных

Первая строка входного файла содержит два числа N и M — количество магазинов и количество льготных маршрутов ($2 \leqslant N \leqslant 500, 0 \leqslant M \leqslant N \cdot (N-1)$). В следующей строке содержатся N разделённых пробелами символов — типы магазинов в том порядке, в котором они занумерованы. Стартовый магазин — единственный, который имеет тип, обозначенный символом «^». Конечный магазин — единственный, которая имеет тип, обозначенный символом «$». Все прочие магазины имеют типы, обозначаемые строчными латинскими буквами.

Следующие M строк содержат описания льготных маршрутов в виде трёх чисел U_i, V_i и P_i — номер магазина, откуда маршрут исходит, номер магазина, куда маршрут входит, и стоимость маршрута ($1 \leqslant U_i, V_i \leqslant N$, $U_i \neq V_i$, $0 \leqslant P_i \leqslant 10^4$). Гарантируется, что каждая пара магазинов встречается в этом списке не более одного раза.

В последней строке содержится непустая строка из строчных латинских букв длиной до 500 символов, которая задаёт последовательность S типов магазинов, которая должна встретиться как подпоследовательность посещённых экспедитором магазинов.

Формат выходных данных

Выведите единственное число — минимальную стоимость поездок. Если требуемого маршрута на такси не существует, выведите «-1».

Пример

стандартный ввод	стандартный вывод
6 12 ^ a b a c $ 1 2 5 1 3 1 2 3 4 3 2 2 2 5 7 5 2 5 3 4 1 4 3 3 4 5 11 5 4 3 2 6 6 4 6 8 abacaba	28

Задача 7. Кубик Рубика (32 МБ)

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	32 мебибайта

Кубик Рубика $N \times N \times N$ представляет собой куб, поверхность которого составлена из $6N^2$ граней малых кубиков, способных вращаться вокруг трёх внутренних осей куба. Каждая грань состоит из N^2 квадратов, окрашенных в один из шести цветов (обозначаются буквами «R», «G», «Y», «O», «B» и «W»). Повороты граней позволяют менять местами цветные квадраты. Например, поворот передней грани состоит в том, что N^2 маленьких кубиков, составляющих переднюю грань, поворачиваются на угол 90, 180 или 270 градусов вокруг оси, проходящей через центр этой грани под прямым углом.

Возможны следующие повороты:

- поворот передней грани (обозначается как «`F`»);
- поворот задней грани (обозначается как «`B`»);
- поворот левой грани (обозначается как «`L`»);
- поворот правой грани (обозначается как «`R`»);
- поворот верхней грани (обозначается как «`U`»);
- поворот нижней грани (обозначается как «`D`»).

По умолчанию имеется в виду поворот по часовой стрелке (если мы смотрим на поворачиваемую грань снаружи кубика). Если поворот делается против часовой стрелки, то такой поворот обозначется той же буквой, но с штрихом (одинарной кавычкой); например поворот передней грани против часовой стрелки обозначается как «`F'`». Поворот на 180 градусов будет обозначаться той же буквой, но с цифрой «`2`»; например поворот передней грани на 180 градусов обозначается как «`F2`».

Заданы некоторые два состояния кубика Рубика: стартовое и целевое. Ваша задача — перевести кубик из стартового состояния в целевое, совершив наименьшее число поворотов.

Формат входных данных

Первая строка входного файла содержит единственное целое число N ($2 \leqslant N \leqslant 3$) — длину ребра кубика Рубика.

Далее заданы две развёртки кубика Рубика: первая соответствует стартовому состоянию, вторая — целевому.

Развёртка задаётся в $3N^2$ строках:

- в первых N строках верхняя грань (ориентирована так, что ребро, примыкающее к передней грани, находится снизу);
- в следующих N строках четыре грани: левая, передняя, правая и задняя (ориентированы так, что рёбра, примыкающие к верхней грани находятся сверху, а примыкающие к нижней грани — снизу);
- и в последних N строках нижняя грань (ориентирована так, что ребро, примыкающее к передней грани, находится сверху).

Формат выходных данных

В первой строке выведите число L — минимальное количество действий, которые нужно выполнить, чтобы привести кубик Рубика из стартового состояния в целевое. В следующих L строках выведите, какие именно действия нужно выполнить. Если для того, чтобы достичь целевое состояние из стартового требуется более восьми поворотов, то в единственной строке выведите число «-1».

Пример

стандартный ввод	стандартный вывод
3 WWO RYY GGG GBYRRRWGBYOO GBYRRRWGBYOO GBRWWWOGBYOO BBB OWW YYR YYY YYY YYY BBBRRRGGGOOO BBBRRRGGGOOO BBBRRRGGGOOO WWW WWW WWW	3 F B2 L
2 RG GO WYRYGWOB BBWBYRGY OR OW BY WG WRBWOGOO BYRYGRGY BR OW	8 U2 L’ U F2 L2 U2 F L’

Задача 8. Съёмки пули

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мебибайт

Смотродел хочет снять очередное видео для своей любимой платформы «Тытруб». Для съёмок у Смотродела заготовлены камера, ружьё и набор из N прямоугольных экранов 1×1. Экраны выстраиваются параллельно одной паре стен комнаты и перпендикулярно полу и другой паре стен таким образом, что все центры экранов лежат на одной прямой AB, перпендикулярной плоскостям экранов. После чего Смотродел встанет слева от всех экранов и произведёт выстрел из ружья в центр самого левого экрана по прямой AB.

Каждый экран характеризуется двумя параметрами, которые Смотродел называет *пробиваемостью* и *вязкостью*. Если пуля достигла экрана с пробиваемостью P и вязкостью $Q \leqslant P$, имея при этом кинетическую энергию E, то:

- если $E < P$, то пуля не пробивает экран, а просто падает перед экраном на пол;
- если $E \geqslant P$, то пуля потеряет кинетическую энергию, равную Q, на преодоление экрана:
  - при этом если $E = Q$, то пуля застрянет в экране и не полетит дальше;
  - а иначе продолжит полёт с кинетической энергией $E - Q$.

Цель Смотродела — снять феноменальное видео, как пуля пробивает все экраны и застревает в самом правом из них. Смотродел может выбрать для установки как все экраны, так и любой непустой их поднабор и расставить выбранные экраны в произвольном порядке.

Помогите Смотроделу: подскажите, какие экраны и в каком порядке поставить, чтобы пуля застряла в самом правом экране. **Минимизировать или максимизировать количество экранов не требуется**.

Потерями кинетической энергии при полёте пули между экранами следует пренебречь.

Формат входных данных

В первой строке записаны два целых числа E и N ($1 \leqslant E \leqslant 30\,000$, $1 \leqslant N \leqslant 10\,000$) — начальную кинетическую энергию пули и количество экранов в наборе, соответственно.

i-я из последующих N строка содержит параметры очередного экрана — целые числа P_i и Q_i ($1 \leqslant P_i \leqslant 30\,000$, $1 \leqslant Q_i \leqslant P$), задающие соответственно пробиваемость и вязкость i-го экрана в наборе.

Формат выходных данных

Если ни при каком выборе экранов не существует расстановки, при которой пуля застрянет в самом правом экране, выведите -1. Иначе в первой строке выведите одно целое число: количество экранов, которые Смотроделу требуется поставить. Во второй строке выведите номера экранов, разделяя их пробелами. Выводить экраны следует слева направо (в том порядке, в котором их будет преодолевать пуля). Если решений несколько, выведите любое.

Пример

стандартный ввод	стандартный вывод
5 1 5 3	-1
4 4 6 4 3 2 1 1 4 2	-1
11 4 3 3 5 4 10 3 7 1	4 3 4 2 1

Задача 9. Сумма делителей

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	3 секунды
Ограничение по памяти:	256 мебибайт

Это интерактивная задача

Загадано целое число n в диапазоне от 1 до 10^6.

Вы можете задавать запросы вида «прибавить к n число q и найти сумму всех различных делителей получившегося числа» ($0 \leqslant q \leqslant 10^6$). Например, при загаданном числе 6 и $q = 0$ ответ будет равен $1 + 2 + 3 + 6 = 12$, а при $q = 1$ число будет равно 7 и ответ будет равен $1 + 7 = 8$.

Ваша задача — угадать число n, задав не более двух запросов.

Протокол взаимодействия

Взаимодействие начинает программа жюри, выводя одно целое число t — количество тестовых примеров ($1 \leqslant t \leqslant 25\,000$) .

В каждом тестовом примере взаимодействие начинает ваша программа, задавая запрос в форме `? q` ($0 \leqslant q \leqslant 10^6$).

В ответ программа жюри выведет одно целое число — сумму всех различных делителей числа $n + q$, где n — загаданное число.

Можно задать не более двух запросов. После этого требуется вывести значение n в форме `! n`.

Если программа жюри вывела `go`, то вы приступаете к взаимодействию в следующем тестовом примере, если она вывела `gg`, ваша программа обязана завершить выполнение.

Пример

стандартный ввод	стандартный вывод
2 12 8 go 1 gg	? 0 ? 1 ! 6 ? 0 ! 1

Задача 10. Обучающая игра (64 МБ)

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	64 мебибайта

Игра «Парсоч для дошкольников» проходит на клетчатом поле $R \times C$.

В центрах некоторых клеток находятся точки. Гарантируется, что количество точек чётно. Между некоторыми парами клеток есть стены.

Известно, что можно разбить точки на пары и соединить их непересекающимися путями, состоящими из отрезков прямых, параллельных сторонам поля, так, чтобы пути не имели общих точек со стенами, а суммарное количество всех клеток, по которым проходят пути, было бы минимальным.

Требуется привести пример любого такого разбиения и соответствующего ему способа соединения точек.

Формат входных данных

В первой строке находятся два целых числа R и C – количество строк и столбцов поля ($1 \leqslant R \leqslant 12$, $1 \leqslant C \leqslant 100$).

В следующей строке находится одно целое число B – количество стен ($0 \leqslant B \leqslant R \cdot (C-1) + C \cdot (R-1) - 1$).

В следующих B строках идут описания стен в виде четырёх целых чисел $r_{1,k}, c_{1,k}, r_{2,k}, c_{2,k}$ – позиции двух соседних клеток поля, между которыми есть стена ($1 \leqslant r_{1,k}, r_{2,k} \leqslant R$, $1 \leqslant c_{1,k}, c_{2,k} \leqslant C$, $(|r_{1,k} - r_{2,k}| + |c_{1,k} - c_{2,k}| = 1$, $1 \leqslant k \leqslant B)$.

Гарантируется, что никакая граница не встречается более одного раза.

В следующей строке находится одно целое число P – количество точек для соединения ($1 \leqslant P \leqslant R \times C$, P — чётное).

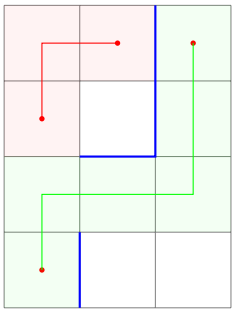
В следующих P строках идут описания клеток, содержащих точки, в виде двух целых чисел r_k, c_k ($1 \leqslant r_k \leqslant R$, $1 \leqslant c_k \leqslant C$, $1 \leqslant k \leqslant P$).

Гарантируется, что все P клеток попарно различны и что на заданном поле можно соединить точки по парам в соответствии с требованиями игры.

Формат выходных данных

В первой строке выведите суммарное количество клеток, по которым проходят пути. В следующих $\frac{P}{2}$ строках для каждого пути вывести его длину и потом все его клетки в порядке следования. Каждая клетка задаётся парой чисел r ($1 \leqslant r \leqslant R$) и c ($1 \leqslant c \leqslant C$), задающей соответственно строку и столбец, которым принадлежит клетка.

Пример

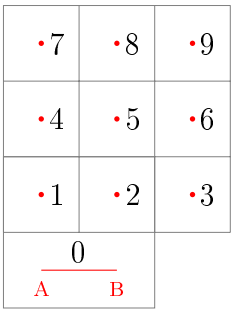
стандартный ввод	стандартный вывод	Пояснение
4 3 4 1 3 1 2 3 2 2 2 2 3 2 2 4 1 4 2 4 4 1 2 1 1 3 1 2	9 3 2 1 1 1 1 2 6 4 1 3 1 3 2 3 3 2 3 1 3	

Задача 11. Одним пальцем 2

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мебибайт

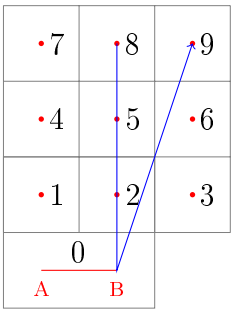
Строка, составленная из цифр от 0 до 9 включительно (возможно, с ведущими нулями) набирается одним пальцем на цифровой клавиатуре.

Клавиши от 1 до 9 имеют размер 1×1 и нажимаются строго в центре. Клавиша 0 имеет размер 2×1 и может быть нажата в любом месте на отрезке AB, соединяющем центры единичных квадратов, образующих эту клавишу.

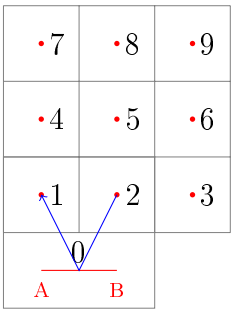


При наборе последовательности цифр палец всегда двигается по кратчайшей траектории.

Например, при наборе числа 809 палец двигается от центра восьмёрки к центру правого квадрата из двух, образующих клавишу 0, затем к центру девятки, так как нажимать клавишу 0 в каких-то точках вне указанного отрезка запрещено.



При наборе числа 201 палец двигается от центра единицы к середине отрезка AB, а затем от него к центру двойки.



Вам дано расстояние, пройденное пальцем, в виде суммы радикалов, то есть суммы элементов вида $a \cdot \sqrt{b}$, где $a \geqslant 0$, $b > 0$ — целые числа, причём не существует $a_1 > a$ и $b_1 < b$ таких, что $a_1 \cdot \sqrt{b_1} = a \cdot \sqrt{b}$.

Выясните, существует ли строка, при наборе которой проходится заданное расстояние, и, если существует, определите минимальное количество цифр в этой строке.

Формат входных данных

Первая строка входных данных содержит одно целое число t — количество тестовых примеров ($1 \leqslant t \leqslant 2 \cdot 10^4$).

Далее следуют тестовые примеры.Первая строка каждого тестового примера содержит одно целое число n ($1 \leqslant n \leqslant 28$) — количество слагаемых в сумме радикалов.

Каждая из последующих n строк содержит по два целых числа a и b ($1 \leqslant a \leqslant 10^6$, $1 \leqslant b \leqslant 42$), задающих слагаемые $a\sqrt{b}$, причём гарантируется, что не существует таких целых положительных a' (не обязательно не превосходящих 10^6) и b', что $a\sqrt{b} = a'\sqrt{b'}$ и $b' < b$.

Формат выходных данных

Выведите одно целое число — минимальное количество цифр в строке, при наборе которой палец в точности проходит расстояние, заданное во входном файле. Если такой строки не существует, выведите -1.

Пример

стандартный ввод	стандартный вывод
2 4 2 1 1 2 3 5 1 10 4 1 1 2 2 3 3 1 41	7 -1

Замечания

В первом тестовом примере возможная строка — `0429490`.

Задача 12. Выбор стороны

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мебибайт

Это интерактивная задача

Загадано римское число заданной длины n. Требуется ответить, можно ли отгадать его менее, чем за n запросов вида «назвать i-ю цифру числа».

В этой задаче вы можете выбрать, на чьей стороне играть — отгадывающего или загадывающего. Если вы считаете, что отгадать число можно, далее вы взаимодействуете с программой жюри, задавая запросы, и после $n-1$ запроса ваша программа обязана назвать число. Если вы считаете, что отгадать число нельзя, далее вы взаимодействуете с программой жюри, отвечая на запросы. При этом после $n-1$ запросов должны остаться как минимум два числа, для которых ответы на соответствующие запросы будут совпадать с теми, что были даны вашей программой.

Напоминаем, что современные римские числа записываются следующим образом (в соответствии с таблицей из Википедии):

Значение разряда	Тысячи	Сотни	Десятки	Единицы
1	M	C	X	I
2	MM	CC	XX	II
3	MMM	CCC	XXX	III
4		CD	XL	IV
5		D	L	V
6		DC	LX	VI
7		DCC	LXX	VII
8		DCCC	LXXX	VIII
9		CM	XC	IX

Правила построения:

- Числа 4, 9, 40, 90, 400 and 900 записываются в реверсивной нотации, где первый символ вычитается из второго (например, для 40 (`XL`) '`X`' (10) вычитается из '`L`' (50)). Это **единственные** места, где реверсивная нотация используется.
- Число, содержащее несколько десятичных цифр, строится дописыванием римского эквивалента каждой цифры от старшего разряда к младшему.
- Если в десятичном разряде стоит 0, никаких цифр в этом разряде в римском представлении не пишется.
- Наибольшее число, которое может быть представлено в римской системе счисления — это число 3 999 (MMMCMXCIX).

Протокол взаимодействия

Взаимодействие начинает программа жюри, выводя одно целое число n ($1 \leqslant n \leqslant 15$).

Далее ваша программа должна выбрать режим игры: 1, если вы готовы угадывать число менее, чем за n попыток, и 0, если вы готовы загадать число, для которого после открытия любых $n-1$ цифр остаётся как минимум два варианта корректного римского числа с этими цифрами.

После этого программа жюри выводит одно целое число g ($1 \leqslant g \leqslant 10^4$) — количество игр, которые вы будете играть в выбранном режиме. Игры являются независимыми, то есть загаданное римское число может меняться между играми (но всегда должно состоять ровно из n римских цифр).

В режиме игры 1 ваша программа задаёт запросы в виде `? p`, где p — позиция цифры в числе, начиная с самой левой ($1 \leqslant p \leqslant n$). В ответ программа жюри выводит соответствующую римскую цифру — одну из букв `I`, `V`, `X`, `L`, `C`, `D` или `M`. Гарантируется, что после любого хода как минимум одно число, в котором все цифры на «открытых» позициях совпадают с теми, что были открыты, существует. Если после $n - 1$ ходов таких чисел более одного, вы проиграли и посылка будет признана неверной. Если вы готовы в какой-то момент однозначно определить число, выведите `! s`, где s — римская запись загаданного числа. Данное действие в общем количестве ходов не учитывается. После вывода ответа сразу же начинается следующая игра (если она есть); в случае, если вы проигрываете, выполнение прерывается и выдаётся соответствующий вердикт.

В режиме игры 0 программа жюри задаёт запросы в виде `? p`, где p — позиция цифры в числе, начиная с самой левой ($1 \leqslant p \leqslant n$). В ответ ваша программа должна выдать римскую цифру, стоящую на соответствующей позиции. Если после какого-то из $n-1$ ходов количество чисел, в которых все цифры на «открытых» позициях совпадают с теми, что были открыты, становится меньше двух, вы проиграли и посылка будет признана неверной. В противном случае после $n - 1$ ходов сразу же начинается следующая игра (если она есть).

Заметим, что в режиме 1 программа жюри достраивает число по мере поступления запросов (но так, чтобы число всегда существовало), то есть интерактор является адаптивным.

Примеры

стандартный ввод	стандартный вывод
3 1 X V	1 ? 1 ? 2 ! XVI
3 2 ? 1 ? 2 ? 2 ? 3	0 X X C C

Замечания

Не забывайте после выбора режима игры, а также после вывода каждого запроса или ответа выводить символ перевода строки и сбрасывать буфер ввода-вывода с помощью вызова функции `flush` используемого языка программирования.