Задача 1. Alpha Protocol

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда 2 секунды (для Java)
Ограничение по памяти:	256 мегабайт

В шпионской ролевой компьютерной игре “Alpha Protocol” есть такая мини-игра для взлома компьютерных терминалов.

На экране показывается таблица размера M на N клеток. Где-то в этой таблице размещено P паролей. Каждый пароль размещается на одной строке и занимает K подряд идущих клеток. Все занимаемые паролями клетки различные, то есть пароли в таблице не могут перекрываться.

В каждой клетке в каждый момент времени показывается какая-то буква. Если клетка занята паролем, то в ней показывается соответствующая буква пароля. Содержимое такой клетки никогда не изменяется. Если же клетка не занята никаким паролем, то в ней показывается произвольная буква, которая может изменяться хоть когда и сколько угодно.

Получается, что большая часть таблицы постоянно изменяется, за исключением тех мест, где расположен пароль. От игрока требуется найти местоположение всех паролей методом пристального вглядывания в таблицу.

В данной задаче известно, что игрок успел разглядеть в таблице S клеток в разные моменты времени. Требуется определить, может ли в этой таблице быть размещено P непересекающихся паролей, и является ли искомое расположение единственно возможным. Если расположение единственное, требуется также найти его.

Учтите, что буквенное содержимое разных паролей может совпадать.

Формат входных данных

В первой строке дано 5 целых чисел: M — количество строк в таблице, N — количество столбцов в таблице, P — количество паролей, K — длина каждого пароля, S — количество известных клеток ($1 \leqslant M, N \leqslant 1\,000$, $1 \leqslant K \leqslant N$, $1 \leqslant K \cdot P \leqslant M \cdot N$, $1 \leqslant S \leqslant 10^6$).

В следующих S строках задана информация об увиденных клетках. В i-ой строке дано два целых числа R_i, C_i и маленькая буква латинского алфавита X_i ($1 \leqslant R_i \leqslant M$, $1 \leqslant C_i \leqslant N$). Это означает, что через i миллисекунд от начала мини-игры игрок увидел, что в клетке таблицы в строке R_i и столбце C_i отображается буква X_i.

Формат выходных данных

В первую строку требуется вывести основной вердикт:

- `impossible` — если корректного размещения паролей в таблице не существует;
- `unique` — если существует ровно одно корректное расположение паролей;
- `ambiguous` — если существует несколько корректных размещений паролей;

Если существует ровно одно возможное расположение, требуется также вывести его в последующих P строках. В каждой строке нужно вывести целые числа R_i и C_i — номер строки и номер столбца самой левой клетки, занятой паролем ($1 \leqslant R_i \leqslant M$, $1 \leqslant C_i \leqslant N$, C_i — минимальное). Выведенные клетки должны быть упорядочены по возрастанию R_i, а при равенстве R_i — по возрастанию C_i.

Примеры

input.txt	output.txt
5 5 1 5 1 3 3 x	ambiguous
1 6 2 3 2 1 3 x 1 3 y	impossible
2 4 4 2 1 1 1 a	unique 1 1 1 3 2 1 2 3
3 5 2 3 16 1 1 c 3 3 q 2 2 z 3 1 p 1 3 a 2 3 x 2 5 z 2 4 y 1 4 d 3 5 a 3 1 s 3 5 b 3 3 q 1 3 b 2 2 a 2 4 y	unique 2 3 3 2

Пояснение к примеру

В первом примере пароль занимает целую строку, и может находиться в любой из строк. Во втором примере пароли должны занимать всю таблицу, что противоречит тому, что в клетке $(1, 3)$ изменилось значение. В третьем примере четыре пароля занимают всю таблицу, по два пароля в строке (`a` — первая буква пароля).

Задача 2. Пифагосфеновы простые

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	4 секунды
Ограничение по памяти:	256 мегабайт

Широко известный в узких кругах древнегреческий математик Пифагосфен изучал простые числа, а также пытался решить задачу об удвоении куба. После очередной неудачной попытки решения Пифагосфен изобрёл новый вид чисел — простые числа Пифагосфена. Целое положительное число является пифагосфеновым простым, если оно, во-первых, является простым, и, во-вторых, представляется в виде суммы кубов двух целых чисел.

Вам даны два целых числа l и r. Определите, сколько пифагосфеновых простых p удовлетворяет неравенству $l \leqslant p \leqslant r$.

Формат входных данных

Первая строка входных данных содержит одно целое число t — количество тестовых примеров ($1 \leqslant t \leqslant 2 \cdot 10^6$).

Каждый тестовый пример состоит из одной строки и содержит два целых числа l и r ($1 \leqslant l \leqslant r \leqslant 10^{10}$).

Формат выходных данных

Для каждого тестового примера выведите одно целое число — количество пифагосфеновых простых чисел между l и r.

Пример

стандартный ввод	стандартный вывод
1 1 10	2

Задача 3. Оптимальный BVH

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

Дано N точек интереса на плоскости. Требуется построить на них BVH-дерево, которое минимизирует среднее время запроса определённого вида.

В BVH-дереве имеется $2N - 1$ узлов, в том числе N листьев. Каждый внутренний узел v дерева имеет ровно двух детей: $\text{left}(v)$ и $\text{right}(v)$.

Каждому из листьев сопоставлена точка интереса взаимно однозначным образом. Будем говорить, что узел v содержит лист u, если v лежит на пути от корня дерева до u. Будем говорить, что узел v содержит точку интереса, если он содержит лист, которому она сопоставлена.

В каждом узле v дерева записан прямоугольник $\text{box}(v)$, выровненный по осям координат. Все точки интереса, которые содержит узел v, лежат внутри или на границе этого прямоугольника, при этом прямоугольник минимально возможный. Например, в любом листе записан вырожденный прямогольник, состоящий из одной точки — точки интереса, которая ему сопоставлена. В корневом узле записан минимальный прямоугольник, содержащий все точки интереса.

Для любой внутренней вершины v прямоугольники $\text{box}(\text{left}(v))$ и $\text{box}(\text{right}(v))$ не пересекаются.

В данной задаче мы будем рассматривать запрос поиска суммы на двумерном префиксе. Запрос Q определяется числами X_q и Y_q. Область этого запроса $\text{area}(Q)$ — это все точки плоскости с координатами x и y, для которых выполнено: $x \leqslant X_q$ и $y \leqslant Y_q$.

Допустим, для каждой точки интереса также указан вес. Ответ на запрос Q — это сумма весов всех точек интереса, которые попадают в область $\text{area}(Q)$. Чтобы быстро отвечать на запросы, в каждом узле BVH-дерева v дополнительно хранится $\text{wsum}(v)$ — сумма весов всех точек интереса, которые этот узел содержит. Запрос обрабатывается следующей рекурсивной функцией, которая вызывается от корня дерева:

```
def solve(Q, v):
    if not intersects(area(Q), box(v)):
        return 0
    elif contains(area(Q), box(v)):
        return wsum(v)
    else:
        return solve(Q, left(v)) + solve(Q, right(v))
```

Временем обработки запроса будем считать выполненное количество рекурсивных вызовов этой функции. Параметры запроса X_q и Y_q генерируются случайным образом независимо друг от друга с равномерным распределением на отрезке $[0 \dots 1]$. Требуется найти такое BVH-дерево, для которого среднее время обработки запроса минимально возможное.

Формат входных данных

В первой строке дано одно целое число N — количество точек интереса ($2 \leqslant N \leqslant 50$). В каждой из оставшихся N строк записано по два вещественных числа x и y — координаты точки интереса ($0 \leqslant x, y, \leqslant 1$). Координаты заданы с не более чем 10 знаками после десятичной точки.

Гарантируется, что все X-координаты различные и все Y-координаты различные.

Формат выходных данных

В первой строке выведите одно вещественное число — искомое среднее время обработки запроса. Это число должно отличаться от точного ответа не более чем на 10^{-12}.

В следующих N строках требуется вывести сопоставление листьев и точек интереса. В каждой v-ой из них содержится одно целое число $\mathrm{index}(v)$ — номер точки интереса, которая сопоставлена листу v. Точки интереса нумеруются начиная с единицы в порядке задания во входном файле.

В оставшихся $N-1$ строках выведите информацию о внутренних узлах дерева. В каждой i-ой из этих строк требуется вывести информацию об узле с номером $v = N + i$ в виде двух положительных целых чисел: $\mathrm{left}(v)$ и $\mathrm{right}(v)$. Оба этих номера должны быть меньше v.

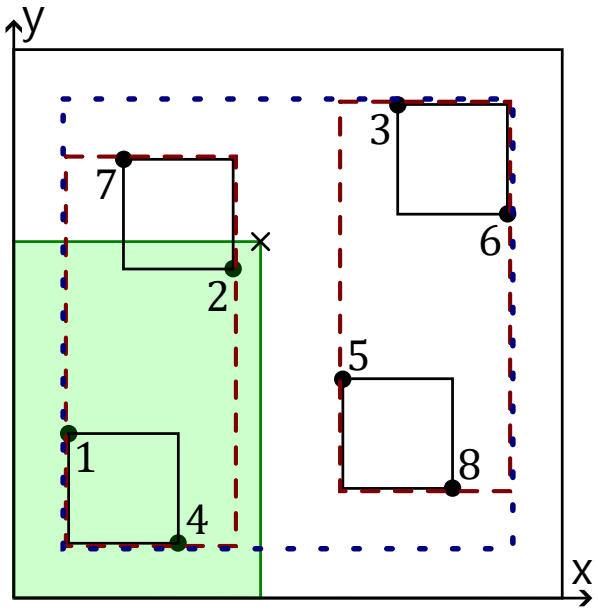
Пример

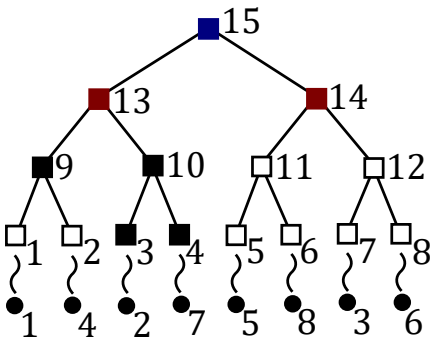
input.txt	output.txt
8 0.1 0.3 0.4 0.6 0.7 0.9 0.3 0.1 0.6 0.4 0.9 0.7 0.2 0.8 0.8 0.2	6.2 1 4 2 7 5 8 3 6 1 2 3 4 5 6 7 8 9 10 11 12 13 14

Иллюстрация

BVH-дерево из примера нарисовано на следующей странице. Справа можно видеть бинарное дерево, листья которого сопоставлены точкам интереса. Слева нарисованы точки интереса и прямоугольники узлов на плоскости.

Слева зелёным цветом отмечен двумерный префикс Q для запроса $X_q = 0.45$, $Y_q = 0.65$. Справа заполненными квадратиками показаны те узлы дерева, для которых вызывается рекурсивная функция. Те узлы, для которых рекурсивная функция не вызывается, показаны пустыми квадратиками. Как видно, время обработки запроса Q равно 7. Для узла 14 срабатывает первое отсечение в функции `Solve`, а для узла 9 — второе, из-за чего функция не вызывается рекурсивно для детей.


y
x
7
2
3
6
5
8
1
4


15
13
14
9
10
11
12
1
2
3
4
5
6
7
8
1
4
2
7
5
8
3
6

Задача 4. Считаем траектории

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секундаы
Ограничение по памяти:	256 мегабайт

Рассмотрим следующие правила построения таблицы ICPC:

- Соревнование длится ровно l минут. Участникам предлагается p задач, пронумерованных с 1 по p.
- На старте соревнования у участника 0 решённых задач и 0 штрафных минут.
- Считается, что если задача сдана в промежутке от x минут 1 секунда до $x+1$ минуты и 0 секунд, включительно, с момента старта соревнования, то задача сдана на $x+1$-й минуте. При этом в 0 минут 0 секунд система ещё не принимает решения, то есть минимальное время сдачи задачи — 0 минут и 1 секунда.
- За задачу, сданную на минуте m, добавляется m минут к штрафному времени.
- За неверные попытки по каждой сданной задаче к штрафному времени добавляется по 20 минут. Неверные попытки по несданным задачам или неверные попытки после сдачи задачи на результат не влияют.
- *Результат* участника — это два целых числа — количество решённых задач и штрафное время на момент завершения соревнования (то есть в момент времени l минут).

Назовём *траекторией* участника список решённых им задач, для каждой из которых зафиксирована минута её сдачи и количество неверных попыток, сделанных по ней до момента сдачи. Более формально траекторию можно определить как последовательность длины $2p$, в которой i-й задаче соответствует или две -1, если задача не сдана, или два целых числа: сначала минута сдачи, а затем — количество неверных попыток до момента сдачи. Из этого определения видно, что попытки по задачам, которые участник не решил, в формировании траектории не участвуют.

Две траектории считаются различными, если соответствующие им последовательности не совпадают.

Вам известно количество задач p, продолжительность контеста l, а также вам дан результат участника — количество решённых задач s и суммарное штрафное время t. Найдите количество различных траекторий, которые могут привести к такому результату. Так как ответ может быть очень большим, выведите остаток от его деления на 998 244 353.

Формат входных данных

Первая строка входных данных содержит четыре целых числа p, l, s и t ($1 \leqslant p \leqslant 10^4$, $1 \leqslant l \leqslant 10^4$, $1 \leqslant s \leqslant p$, $1 \leqslant t \leqslant 2 \cdot 10^4$).

Формат выходных данных

Выведите одно целое число — остаток от деления количества различных траекторий, которые могут привести к заданному результату,

Примеры

стандартный ввод	стандартный вывод
13 300 1 22	26
13 300 9 8	0
13 300 2 22	1794

Задача 5. Прыжки

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

Это интерактивная задача

Алиса находится в центре круга заданного радиуса R. Она хочет выбраться из этого круга. Для этого она играет с Бобом в следующую игру:

Задан массив из n целых положительных чисел x_i $(1 \leqslant i \leqslant n)$. На каждом шаге Алиса выбирает произвольное направление (указывает вектор длины 1 с вещественными координатами). Далее Боб либо оставляет этот вектор, либо меняет его направление на противоположное. Затем в выбранном направлении на i-м шаге Алиса перемещается ровно на расстояние x_i. Если Алисе удастся за не более, чем n шагов выбраться из круга — она выигрывает, если нет — выигрывает Боб.

Ваша цель — определить, кто из двух игроков выиграет, и сыграть за него.

Протокол взаимодействия

Взаимодействие начинает программа жюри, выводя в первой строке входных данных два целых числа R и n — радиус круга и количество шагов $(1 \leqslant R \leqslant 10^5, 1 \leqslant n \leqslant 10^3)$.

Во второй строке заданы n положительных целых чисел, не превосходящих 10^3 — длины шагов.

После этого вы сообщаете жюри, за кого вы будете играть — за Алису или за Боба. Для этого нужно выдать в выходной поток либо строку `! Alice` , либо `! Bob`. Если вы играете за Алису, то на каждом шаге вы указываете направление своего прыжка (выдаете в выходной поток два вещественных числа, образующих вектор единичной длины с точностью 10^{-6}).

Далее вы считываете ответ Боба, за которого играет программа жюри, — либо `+` либо `-`, указывающие соответственно, что Боб оставил ваше направление либо поменял его на противоположное.

Игра заканчивается, если на каком-то шаге Алисе удалось выйти за пределы круга.

Если же вы играете за Боба, то, напротив, на каждом шаге вы сначала считываете выбранное Алисой (программой жюри) направление, а в ответ выдаете либо `+` либо `-`.

Гарантируется, что изменение расстояния до центра круга на 10^{-6} никак не отразится на ходе игры.

Примеры

стандартный ввод	стандартный вывод
5 1 7 +	 ! Alice 1 0
5 2 2 2 1 0 1 0	 ! Bob + +

Пояснение к примерам

В первом примере, куда бы ни пошла Алиса из центра круга радиуса 5, она в любом случае за один шаг длины 7 выберется из круга.

Во втором примере, как бы ни ходила Алиса, что бы ни предпринимал Боб, за два хода длины 2 каждый, Алиса никак не сможет уйти дальше, чем на расстояние длины 4 от центра круга. Следовательно, она не сможет выйти за пределы круга радиуса 5.

Задача 6. Меч или магия

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	3 секунды
	6 секунд (для Java)
Ограничение по памяти:	256 мегабайт

Мстислав играет в очередную компьютерную игру. Ему известно N вариантов экипировки персонажа и M типов монстров. Он хочет узнать, в каких случаях эффективнее атаковать монстра мечом, а в каких — магией.

Допустим, базовый урон меча равен D, а защита монстра равна A. Защита определяет, сколько единиц урона поглощает броня. Если $D > A$, тогда удар мечом наносит урон $D - A$. Однако если $D \leqslant A$, тогда удар мечом не наносит никакого урона.

Некоторые мечи частично или полностью игнорируют защиту, в этом случае задана доля игнорирования P процентов. Чтобы вычислить урон от удара таким мечом, нужно:

1. вычислить урон X с учётом защиты монстра,
2. вычислить урон Y по той же схеме, но считая защиту монстра нулевой,
3. сложить P процентов от числа Y и $(100 - P)$ процентов от числа X.

Примеры вычисления можно посмотреть в иллюстрации к примеру.

Кроме того, персонаж может атаковать магией. Допустим, базовый урой магией равен S, а монстр имеет сопротивление магии R процентов. Тогда атака магией наносит урон, равный $(100 - R)$ процентов от числа S.

Требуется для каждого варианта экипировки определить, сколько типов монстров таковы, что атака магией наносит строго больше урона, чем мечом. Аналогично, для каждого типа монстра определите, сколько вариантов экипировки таковы, что атака магией сильнее.

Формат входных данных

В первой строке дано два целых числа: N — количество вариантов экипировки, M — количество типов монстров ($1 \leqslant N, M \leqslant 10^5$).

В следующих N строках заданы варианты экипировки. В каждой i-ой строке дано три целых числа: D_i — базовый урон мечом, P_i — доля игнорирования защиты в процентах, S_i — базовый урон магией ($0 \leqslant D_i, S_i \leqslant 10^9$, $0 \leqslant P_i \leqslant 100$).

В последних M строках заданы типы монстров. В каждой j-ой строке дано два целых числа: A_j — защита, R_j — сопротивление магии в процентах ($0 \leqslant A_j \leqslant 10^9$, $0 \leqslant R_j \leqslant 100$).

Формат выходных данных

В первых N строках выведите ответы для вариантов экипировки. Для каждой экипировки в порядке задания во входном файле выведите, сколько есть типов монстров, по которым атака магией сильнее.

В следующих M строках выведите ответы для типов монстров. Для каждого монстра в порядке задания во входном файле выведите, сколько есть вариантов экипировки, с которыми атака магией по этому монстру сильнее.

Пример

input.txt	output.txt
4 6 10 0 3 3 0 10 5 50 5 6 80 10 0 0 5 0 10 0 5 50 10 50 5 100	2 5 2 4 2 3 4 1 3 0

Пояснение к примеру

Рассмотрим подробнее вычисления для последней экипировки.

Меч имеет базовый урон 6 и 80% игнорирования защиты, второй монстр имеет защиту 5. Тогда урон с учётом защиты равен $X = 1$, а урон без учёта защиты равен $Y = 6$. Реальный урон мечом составляет $80\% * 6 + 20\% * 1 = 5$. Аналогично для четвёртого и шестого монстров.

У третьего и пятого монстров защита выше базового урона меча, так что X снижается до нуля, и окончательный урон равен 4.8. У первого монстра нет защиты, так что меч наносит полный урон 6.

С другой стороны, атака магией наносит полный урон 10 первым трём монстрам, 5 единиц урона четвёртому и пятому монстрам, и нулевой урона последнему монстру. Получается, что с последней экипировкой атака магией слабее атаки мечом против пятого монстра, и равна по силе атаке мечом против четвёртого монстра. Для остальных четырёх монстров атака магией сильнее.

Задача 7. Звёздочки

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

Вам дано целое положительное число, записанное в десятичной системе счисления без ведущих нулей.

Требуется найти остаток от деления на 998 244 353 количества способов заменить в этом числе цифры звёздочками так, что ни одно из целых положительных чисел без ведущих нулей, которое будет получаться заменой звёздочки какой-либо цифрой, не делится на 11.

Формат входных данных

Входные данные состоят из одного целого числа n, записанного без ведущих нулей ($1 \leqslant n < 10^{100\,000}$).

Формат выходных данных

Выведите одно целое число — остаток от деления количества способов заменить в заданном числе цифры звёздочками так, что ни при какой замене звёздочек цифрами не получится целого положительного числа без ведущих нулей, делящегося на 11.

Примеры

стандартный ввод	стандартный вывод
8	1
23	0

Задача 8. Художники

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

В погоне за стульями Остап Бендер и Ипполит Матвеевич Воробьянинов проникли на пароход «Скрябин» под видом художника и его помощника. Первым заданием было украсить пароход красными звёздами.

Процесс рисования звёзд выглядит следующим образом: Остап отмечает серым карандашом на окружности единичного радиуса n точек с полярными координатами $(1, 2 \cdot i \cdot \pi/n)$ для i от 0 до $n - 1$. Затем Ипполит Матвеевич выбирает какую-то из точек и выполняет следующие действия:

- Красит выбранную точку в красный цвет.
- Находит точку, в которую выбранная точка переходит при повороте окружности на $2 \cdot k \cdot \pi/n$ по часовой стрелке относительно её центра.
- Соединяет выбранную точку с этой точкой отрезком, проводя линию серым карандашом.
- Если новая точка окрашена в красный, процесс завершается. Иначе новая точка становится выбранной и процесс продолжается.
- Если два нарисованных отрезка пересеклись во внутренней точке, точка пересечения окрашивается в красный цвет.

По заданным n и k определите, сколько точек будет в итоге окрашено в красный цвет.

Формат входных данных

В перваой строке входных данных записано одно целое число t — количество тестовых примеров ($1 \leqslant t \leqslant 10^4$).

Каждая из последующих t строк содержит по два целых числа n и k, соответствующих одному тестовому примеру ($3 \leqslant n \leqslant 10^3$, $1 \leqslant k < n$).

Формат выходных данных

Для каждого тестового примера необходимо вывести в отдельную строку одно целое число — количество точек, окрашенных в красный цвет.

Пример

стандартный ввод	стандартный вывод
3 5 2 4 2 3 1	10 2 3

Задача 9. Система контроля задач

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

Известный программист Василий придумал и разработал уникальную систему управления задачами, которая позволяет эффективно контролировать и модифицировать решения. В его системе присутствует набор из n задач, каждая из которых имеет определённое решение, представленное в виде последовательности из a_i строк.

Его система поддерживает следующие основные операции:

- `change` i x — заменить длину решения i-й задачи на значение x;
- `print` i — вывести на экран длину решения i-й задачи;
- `checkpoint` *name* — создать именованную контрольную точку с уникальным именем *name*, сохраняя текущее состояние всех задач;
- `rollback` *name* — восстановить состояние задач до момента создания контрольной точки *name*, удаляя все последующие контрольные точки;
- `commit` *name* — удалить все контрольные точки, созданные ранее контрольной точки *name*.

Кроме этого, Василий предусмотрел ограничения:

- При создании контрольной точки имя должно быть уникальным. Оно не должно использоваться ранее ни в одной контрольной точке, даже в уже удалённой.
- При выполнении операций `rollback` и `commit` указанная контрольная точка должна существовать и не быть удалённой ранее.
- Между двумя контрольными точками должно быть хотя бы одно изменение. Если создать новую контрольную точку без изменений после предыдущей, то она удалит предыдущую.

К сожалению, компьютер Василия был утерян, а вместе с ним и реализация этой системы. Василий сейчас занят, поэтому попросил Вас воссоздать функциональность данной системы контроля задач, реализовав все описанные операции и ограничения. Сможете ли Вы?

Формат входных данных

В первой строке дано два целых числа n и m — количество задач в системе и количество операций с системой ($1 \leqslant n, m \leqslant 10^5$).

Во второй строке содержатся n целых чисел a_i — длины решений задач ($1 \leqslant a_i \leqslant 10^9$).

В следующих m строках содержатся операции с системой:

- `change` i x сделать a_i равным x ($1 \leqslant i \leqslant n, 1 \leqslant x \leqslant 10^9$).
- `print` i — вывести на экран длину решения i-й задачи ($1 \leqslant i \leqslant n$).
- `checkpoint` *name* — создать контрольную точку с именем *name*.

- `rollback` *name* — вернуться к контрольной точке *name*, удаляя все контрольные точки, созданные позднее.
- `commit` *name* — удалить все контрольные точки, созданные ранее контрольной точки *name*.

name во всех операциях состоит из строчных букв латинского алфавита, и имеет длину не менее 1 и не более 30.

Формат выходных данных

Для каждой операции над системой нужно вывести сообщение с результатом выполнения:

- `change` — требуется вывести строку “`OK`”. Предполагаем, что эта операция всегда выполняется успешно.
- `print` — требуется вывести значение a_i.
- `checkpoint` — требуется вывести строку “`OK`”, если контрольная точка создалась успешно, и строку “`ERROR`” в противном случае.
- `rollback` — требуется вывести строку “`OK`”, если контрольная точка существует, и строку “`ERROR`” в противном случае.
- `commit` — требуется вывести строку “`OK`”, если контрольная точка существует, и строку “`ERROR`” в противном случае.

Примеры

стандартный ввод	стандартный вывод
5 13 1 2 3 4 5 print 1 change 1 10 print 1 checkpoint first change 1 20 print 1 rollback first print 1 change 1 20 checkpoint first checkpoint second commit second rollback first	1 OK 10 OK OK 20 OK 10 OK ERROR OK OK ERROR

Задача 10. Добрые числа

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
	2 секунды (для Java)
Ограничение по памяти:	256 мегабайт

На уроке истории Алиса и Боб играют в следующую игру: Алиса загадывает целое число от 1 до 3999 и записывает его римскими цифрами. Боб хочет угадать это число. За один ход он может предложить произвольную строку из букв `IVXLCDM` и узнать, будет загаданное число Алисы корректным при дописывании этой строки спереди (или сзади — на выбор Боба). Отметим, что каждый раз дописывание идёт к задуманному числу — фактически задуманная строка не меняется.

Если загаданное число можно определить однозначно за конечное количество запросов, то Алиса называет это число *добрым*.

Вам задано целое число n. Найдите самое близкое к нему «доброе» число. Если таких чисел несколько, выведите наименьшее из них.

Напоминаем способ записи римских чисел (таблица взята 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

Заметим, что:

- Числа 4, 9, 40, 90, 400 и 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).

Формат входных данных

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

Каждая из последующих t строк содержит одно целое число n ($1 \leqslant n \leqslant 10^4$).

Формат выходных данных

Для каждого n в отдельной строке выведите одно целое число — наименьшее из самых близких к n «добрых» чисел.

Пример

стандартный ввод	стандартный вывод
2 3 8	3 7

Задача 11. Бегство от паука

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	2 секунды
	4 секунды (для Java)
Ограничение по памяти:	256 мегабайт

Игрок убегает от гигантского паука в лабиринте. Лабиринт состоит из N перекрёстков и M туннелей, соединяющих перекрёстки. Для каждого туннеля известна длина в метрах. По туннелю можно перемещаться в обе стороны.

Игрок бежит по лабиринту со скоростью U метров в секунду, а паук — со скоростью V метров в секунду. Игрок начинает свой путь в перекрёстке A и должен попасть в перекрёсток B, где расположен выход из лабиринта. Паук начинает бежать из того же перекрёстка A спустя T_0 секунд после того, как там появляется игрок.

Между пауком и игроком протягивается нить паутины, которая начинается во рту паука и приклеена к игроку. Когда игрок бежит, паутина удлиняется вдоль пробегаемого им пути. Паук действует инстинктивно и просто поедает свою паутину с другого конца. Паук съедает игрока только в том случае, если паутина полностью заканчивается. Если паук и игрок оказываются в одной точке, но между ними ещё есть паутина, паук не замечает игрока и продолжает поедать паутину. Если в местоположении паука уже проложено несколько нитей паутины, паук продолжает есть ту нить, которая непосредственно выходит из его рта, он никогда не перескакивает между нитями паутины.

В некоторых перекрёстках лабиринта лежат зелья ускорения. Когда игрок пробегает через такой перекрёсток, он берёт зелье ускорения и сразу же выпивает его. Когда игрок выпивает зелье ускорения, его скорость увеличивается до W на следующие T_p секунд. А если игрок уже находится под действием зелья ускорения в этот момент, то длительность ускорения увеличивается на T_p секунд.

Требуется определить, может ли игрок добраться до выхода из лабиринта до того, как его съест паук, и вывести путь, если это возможно. Если паутина заканчивается ровно в тот момент, когда игрок добегает до выхода, тогда паук съедает игрока.

Формат входных данных

В первой строке дано пять целых чисел: N — количество перекрёстков, M — количество туннелей, K — количество зелий скорости, A — номер перекрёстка, в котором начинают игрок и паук, B — номер перекрёстка, где расположен выход из лабиринта ($2 \leqslant N \leqslant 30\,000$, $0 \leqslant M \leqslant 30\,000$, $1 \leqslant K \leqslant 20$, $1 \leqslant A, B \leqslant N$, $A \neq B$).

Во второй строке дано три целых числа: U — нормальная скорость игрока, V — скорость паука, W — скорость игрока, когда действует зелье ускорения ($1 \leqslant U < V < W \leqslant 10^5$).

Во третьей строке дано два целых числа: T_0 — время между стартом игрока и стартом паука, T_p — сколько времени действует зелье ускорения ($1 \leqslant T_0, T_p \leqslant 10^6$).

В четвёртой строке дано K различных целых чисел от 1 до N включительно — номера перекрёстков, в которых расположены зелья скорости.

В оставшихся M строках описаны туннели лабиринта. Каждый туннель задан тремя целыми числами i, j, L, которые означают, что между перекрёстками i и j есть туннель длиной L ($1 \leqslant i < j \leqslant N$, $1 \leqslant L \leqslant 10^9$). Одну и ту же пару перекрёстков может напрямую соединять несколько туннелей.

Формат выходных данных

Если игрок не может убежать от паука, выведите лишь одно число -1.

Если игрок может добежать до выхода, выведите спасительный путь. Сначала выведите Q — через сколько туннелей нужно пробежать ($1 \leqslant Q \leqslant 10^6$), а затем Q целых чисел: номера туннелей, по которым игроку следует бежать, в хронологическом порядке. Туннели нумеруются от 1 до M в порядке задания во входном файле.

Если спасительных путей несколько, разрешается вывести любой из них.

Примеры

input.txt	output.txt
3 3 1 1 3 10 20 100 2 100 3 1 2 10 1 2 20 2 3 30	-1
3 2 3 1 3 10 50 100 2 1 1 2 3 1 2 50 2 3 199	2 1 2
5 4 2 1 3 10 20 40 2 10 4 5 1 2 10 2 3 300 2 4 20 2 5 100000	4 1 3 3 2

Пояснение к примерам

В первом примере единственное зелье ускорения лежит на выходе из лабиринта, и потому оно бесполезно. Оптимальный путь игрока — бежать сначала по туннелю 1 до перекрёстка 2, а затем по туннелю 3 до выхода. Однако этот путь имеет длину 40 метров, и игроку требуется 4 секунды, чтобы его пробежать. Паук в два раза быстрее и пробежит этот путь за 2 секунды. Начальная фора игрока составляет 2 секунды, так что паук и игрок окажутся в выходном перекрёстке лабиринта одновременно.

Во втором примере есть только один путь до выхода, и игрок едва успевает убежать. Благодаря зелью ускорения игрок оказывается в перекрёстке 2 через полсекунды после начала (паук ещё не стартовал). Он выпивает второе зелье и имеет полторы секунды ускорения в запасе, что позволяет ему покрыть первые 150 метров туннеля 2 ровно к моменту старта паука. Оставшиеся 49 метров игрок пробегает за 4.9 секунды, а пауку нужно 4.98 секунд, чтобы преодолеть 249 метров.

В последнем примере игрок бежит по перекрёсткам 1, 2, 4, 2, 3. Ему приходится забежать в перекрёсток 4 за зельем ускорения, которого хватает на весь оставшийся путь. На обратном пути по туннелю 3 он встречается с пауком, однако паук слишком глуп и не замечает этого.

Задача 12. Производство треугольников

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

Инженер Брунс устроился на завод по производству треугольников. На склад завода поступают стороны от двух фабрик. Стороны от Старгородской фабрики имеют длину a, стороны Арбатовской фабрики — длину b. Производственный процесс заключается в том, что выбираются три стороны со склада и из них делается треугольник ненулевой площади.

Инженер Брунс хочет понять, сколько разных моделей треугольников может выпускать завод, используя поставляемые стороны. Две модели треугольников считаются одинаковыми, если треугольники этих моделей можно перевести друг в друга движением плоскости.

Формат входных данных

Первая строка входных данных содержит одно целое число a — длину стороны, выпускаемой Старгородской фабрикой ($1 \leqslant a \leqslant 10^4$).

Вторая строка содержит одно целое число b — длину стороны, выпускаемой Арбатовской фабрикой ($1 \leqslant b \leqslant 10^4$).

Формат выходных данных

Выведите одно целое число — количество моделей треугольников, которые может выпускать завод.

Примеры

стандартный ввод	стандартный вывод
1 1	1
20 25	4