Problem A. CosmoCraft

Input file:	`cosmo.in`
Output file:	`cosmo.out`
Time limit:	3 seconds
Memory limit:	256 mebibytes

В предназначенной для двух игроков игре CosmoCraft требуется управлять экономикой для того, чтобы построить победоносную армию. Юниты в игре подразделяются на три типа — гражданские объекты, заводы и военные объекты. Игра проходит в пошаговом режиме.

- Каждый гражданский объект даёт вам прибыль размером 1 космобакс в ход.
- Каждый завод даёт вам возможность за 1 космобакс и 1 ход построить либо гражданский объект, либо военный объект, на ваш выбор.
- Постройка завода стоит 1 космобакс.
- Военные объекты могут быть использованы для разгрома ваших оппонентов.

Вы начинаете игру с n гражданскими объектами и k заводами. На каждом ходу вы можете потратить полученный с гражданских объектов доход на создание гражданских объектов, заводов, военных объектов. При этом строительство занимает один ход (то есть «заказанные» объекты входят в строй к следующему ходу. Непотраченная сумма остаётся на следующий ход.

В конце хода вы можете собрать некоторое количеств военных объектов и атаковать оппонента. Если размер вашей армии (количество выделенных вами для атаки военных объектов) строго больше количества размера армии (общего количества военных объектов) у оппонента, оппонент проигрывает игру, а у вас остаётся количество военных объектов, равное разности между размерами вашей и его армии. В ином случае ваша армия оказывается разгромлена, а у оппонента остаётся количество военных объектов, равное разности между размерами вашей и его армии, при этом проигравшим игру (а не битву) вы не считаетесь. Игра заканчивается после t ходов, и обычно по завершении этих ходов оппоненты обмениваются ударами «в полную силу»

Вы играете против хорошо изученного оппонента, и вы знаете, на каком ходу какими силами он атакует. Вы приняли решение сыграть «от обороны» — отразить все атаки оппонента, а на последнем ходу нанести контрудар.

Ваша задача — найти максимальный размер армии, которая у вас будет собрана к последнему ходу.

Input

Входной файл состоит из нескольких (не более, чем 50) тестовых примеров. Каждый тестовый пример начинается со строки, содержащей три целых числа: n, k и t, где n ($1 \le n \le 100$) — количество ваших гражданских объектов перед началом игры, k ($1 \le k \le 100$) — количество заводов перед началом игры и t ($1 \le t \le 10,000$) — количество ходов в партии. В следующей строке задано $t-1$ целое число a_i ($0 \le a_i \le 2^{63}-1$). i-е из них задаёт силу атаки (количество военных единиц, выделенных вашим оппонентом для атаки) на i-м ходу. Входной файл заканчивается примером с $n = k = t = 0$, обрабатывать который не надо.

Output

Для каждого тестового примера выведите в отдельной строке целое число — максимальный размер вашей армии в конце игры. Выведите -1, если отразить все атаки оппонента невозможны. Гарантируется, что для всех тестовых примеров ответ не будет превосходить $2^{63}-1$.

Example

cosmo.in	cosmo.out
8 4 6 22 6 10 14 0 4 3 3 0 0 6 9 7 0 0 11 0 7 0 0 0 0	-1 11 101

Problem B. Covered Walkway

Input file:	`covered.in`
Output file:	`covered.out`
Time limit:	30 seconds
Memory limit:	256 Mebibytes

После зимнего сезона в некоторых точках автомагистрали оказался разрушен асфальт. Поэтому было принято решение поменять покрытие дороги хотя бы частично, но так, чтобы в местах, где асфальт разрушен, оказалось новое покрытие.

Расценки компании-подрядчика выглядят следующим образом: стоимость замена покрытия дороги между двумя точками на расстоянии x и y от начала магистрали равна $c+(x-y)^2$, где c — константа, определённая подрядчиком заранее. В случае, если меняется покрытие дороги в одной точке, стоимость работ, таким образом, будет равна c.

По заданным координатам точек с разрушенным асфальтом вычислите, какова минимальная стоимость смены покрытия дороги, после которой во всех этих точках асфальт будет восстановлен.

Input

Входной файл содержит несколько тестовых примеров (не более 15). Каждый тестовый пример начинается со строки, содержащей два целых числа n ($1 \le n \le 10^6$) и c ($1 \le c \le 10^9$), где n — количество точек, в которых покрытие разрушено, а c — определённая подрядчиком константа. Каждая из последующих n строк содержит одно целое положительное число, задающее расстояние от начала магистрали до очередной точки с разрушенным покрытием. Точки задаются в порядке следования от начала магистрали, значение расстояния от точки до начала магистрали не превосходит 10^9. Входной файл завершается тестовым примером с $n = c = 0$, обрабатывать который не надо.

Output

Для каждого тестового примера выведите в отдельной строке одно целое число — минимальную стоимость смены покрытия дороги, после которой дорожное покрытие не будет содержать "разрушенных" точек.

Example

covered.in	covered.out
10 5000 1 23 45 67 101 124 560 789 990 1019 0 0	30726

Problem C. Double Dealing

Input file:	doubledealing.in
Output file:	doubledealing.out
Time limit:	25 seconds
Memory limit:	256 mebibytes

Рассмотрим колоду, состоящую из n попарно различных карт.

Шаг перетасовки состоит из двух действий: первым действие разделим эту колоду на k кучек следующим образом: верхняя карта идёт в первую кучку, следующая — во вторую, и так далее (номер кучки, в которую идёт i-я карта, есть увеличенный на единицу остаток от деления i на k). Внутри каждой кучки каждая новая карта кладётся на предыдущую.

Вторым действием собираем колоду следующим образом — сначала кладём k-ю кучку, на неё складываем $k-1$-ю и так далее, то есть первая кучка будет сверху, а k-я снизу.

За какое наименьшее положительное количество шагов перетасовки можно добиться того, что колода окажется в первоначальном состоянии?

Input

Входной файл состоит из нескольких тестовых примеров. Каждый тестовый пример представляет собой одну строку, содержащую два целых числа n и k ($1 \le n \le 800$, $1 \le k \le 800$). Никакие два тестовых примера во входном файле не совпадают. Входной файл заканчивается строкой, содержащей два нуля, обрабатывать которую не требуется.

Output

Для каждого тестового примера в отдельной строке выведите целое число — количество шагов перетасовки, требуемое для того, чтобы вернуть колоду карт в первоначальное состояние.

Example

doubledealing.in	doubledealing.out
1 3 10 3 52 4 0 0	1 4 13

Problem D. Баянчик

Input file:	endofworld.in
Output file:	endofworld.out
Time limit:	2 seconds
Memory limit:	256 mebibytes

Легенды многих задач говорят о том, что где-то в Азии есть группа монахов, которая всё ещё решает головоломку "Ханойская башня". Для тех, кто встречается с подобной задачей впервые, напоминаем, что "Ханойская башня" представляет собой три стержня, на которых можно размещать диски попарно различных радиусов. Изначально все диски расположены на одном из стержней так, что диск с меньшим радиусом всегда лежит на диске с большим радиусом. Требуется перенести все диски на другой стержень так, чтобы в процессе решения диск с большим радиусом ни разу не оказался положен на диск с меньшим радиусом. При этом переносить диски можно только по одному.

Вам дано состояние головоломки в момент, когда монахи решили отвлечься на просмотр финала Лиги Чемпионов. Вычислите, сколько ходов (переносов дисков) осталось сделать монахам до момента, когда головоломка будет решена в случае, если те будут действовать оптимальным образом.

Input

Входной файл состоит из нескольких (не более, чем 2500) тестовых примеров. Каждый тестовый пример состоит из строки длиной не менее одного и не более 63 символов, содержащей только буквы `A`, `B` и `C`. Длина строки задаёт количество дисков, а каждый символ задаёт номер стержня, на котором находится диск, при этом диски перечислены по возрастанию их радиусов (первый символ соответствует диску с наименьшим радиусом, последний — с наибольшим). Изначально все диски были на стержне с номером `A`, требуется переместить их на стержень с номером `B`. Гарантируется, что заданная позиция может возникнуть при решении исходной задачи оптимальным способом. Входной файл завершается строкой, содержащей единственный символ — заглавную букву `X`.

Output

Для каждого тестового примера в отдельной строке выведите одно целое число — минимальное число ходов, которое осталось сделать монахам для того, чтобы головоломка была решена.

Example

endofworld.in	endofworld.out
AAA BBB X	7 0

Problem E. Estimation

Input file:	estimate.in
Output file:	estimate.out
Time limit:	16 seconds
Memory limit:	256 mebibytes

Задан массив A, состоящий из целых чисел. "Приближение" массива A строится следующим способом: массив делится на k непрерывнах блоков (блоки не обязаны иметь равную длину), после чего строится массив B такой, что если i и j-й элементы находятся в одном блоке, то $B[i] = B[j]$. Требуется построить приближение с наименьшей ошибкой. Ошибка приближения вычисляется по формуле $(\sum |A[i] - B[i]|)$.

Input

Входной файл состоит из нескольких (25 или менее) тестовых примеров. Каждый тестовый пример начинается со строки, содержащей два целых числа n $(1 \le n \le 2,000)$ и k $(1 \le k \le 25,\ k \le n)$, — размер массива A и количество блоков, на которое массив требуется разбить в процессе приближения. В n последующих строках задан массив A, перечисленный от начального элемента к конечному; в каждой строке задано одно целое число, не превосходящее $10,000$ по абсолютной величине. Заканчивается входной файл тестовым примером с $n = k = 0$, обрабатывать который не следует.

Output

Для каждого тестового примера в отдельной строке выведите одно целое число — значение минимальной ошибки при приближении.

Example

estimate.in	estimate.out
7 2 6 5 4 3 2 1 7 0 0	9

Problem F. Juggler

Input file:	juggler.in
Output file:	juggler.out
Time limit:	4 seconds
Memory limit:	256 mebibytes

Во время циркового представления артист жонглирует одной рукой некоторым количеством занумерованных шаров, летающих по цепочке.

В конце номера он должен выбросить в корзину шары в порядке возрастания их номеров. Жонглёр может совершать следующие действия: «сдвинуть» объекты на один по часовой стрелке (то есть переместить руку по ходу вращения), «сдвинуть» объекты на один против часовой стрелки (то есть переместить руку против хода вращения), или же положить оказавшийся в руке шар с нужным номером в корзину, после чего в его руке оказывается шар, следующий за выброшенным по часовой стрелке. Каждое из трёх действий считается одним ходом. За какое минимальное количество ходов жонглёр сможет завершить свой номер?

Input

Входной файл содержит несколько (не более 30) тестовых примеров. В первой строке каждого тестового примера задано целое число n, ($1 \le n \le 10^5$) — количество шаров, используемых при жонглировании. Каждая из последующих n строк содержит целое число k_i ($1 \le k_i \le n$) — номер, написанный на i-м шара (отсчёт производится, начиная с единицы, по часовой стрелке от шара, который находится в настоящий момент в руке жонглёра). Гарантируется, что набор чисел k_i является перестановкой чисел $1 \dots n$. Входной файл заканчивается тестовым примером с $n = 0$, который обрабатывать не требуется.

Output

Для каждого тестового примера в отдельной строке выведите целое число — минимальное количество ходов, требующихся жонглёру для завершения номера.

Example

juggler.in	juggler.out
3 3 2 1 0	5

Note

В приведённом примере в руке жонглёра находится шар, который должен оказаться в корзине третьим, следующим по часовой стрелке идёт шар, который должен оказаться в корзине вторым, следующим по часовой стрелке идёт шар, который должен оказаться в корзине первым. Соответственно, жонглёр смещается против часовой стрелки, выбрасывае первый шар (получает в руку третий), смещается по часовой стрелке, выбрасывает второй и выбрасывает третий — итого 5 ходов.

Problem G. Red-Blue Spanning Tree

Input file:	`redblue.in`
Output file:	`redblue.out`
Time limit:	10 seconds
Memory limit:	256 mebibytes

Дан неориентированный связный граф, в котором каждое ребро окрашено в один из двух цветов — синий или красный. Определите, существует ли остовное дерево, содержащее ровно k синих рёбер.

Input

Входной файл состоит из нескольких (не более 40) тестовых примеров. Первая строка каждого тестового примера содержит по три целых числа n, m и k, где n ($2 \le n \le 1000$) — количество вершин в графе, m — количество рёбер и k ($0 \le k < n$) — требуемое количество синих рёбер в остовном дереве.

Каждая из последующих m строк содержит три элемента, описывающие рёбра: c, f, t, где c — '`R`', если ребро окрашено в красный цвет, и '`B`' в противном случае (то есть если ребро синее), а f и t — целые числа ($1 \le f, t \le n, t \ne f$), обозначающие номера вершин, соединённых ребром. Гарантируется, что граф является связным, и что любые две вершины соединены не более, чем одним ребром.

Входной файл завершается тестовым примером с $n = m = k = 0$, который обрабатывать не требуется.

Output

Для каждого тестового примера в отдельной строке выведите 1, если возможно построить остовное дерево ровно с k синими рёбрами и 0 в противном случае.

Example

redblue.in	redblue.out
3 3 2 B 1 2 B 2 3 R 3 1 2 1 1 R 1 2 0 0 0	1 0

Problem H. The Red Gem (Division 1 Only!)

Input file:	`redgem.in`
Output file:	`redgem.out`
Time limit:	2 seconds
Memory limit:	256 mebibytes

В стране Круганде в музее кругов выставлена уникальная достопримечательность — красный круглый алмаз. Алмаз помещён на фиолетовую круглую платформу вместе с обычными в Круганде оранжевыми круглыми алмазами.

После неурожая в Круганде царит голод, так что жители страны отощали и из кругов стали точками. Учитывая, что вход на фиолетовую платформу запрещён, жители могут располагаться только по её окружности. К сожалению, с некоторых точек красный круглый алмаз не виден или виден плохо, так как вид закрывается оранжевыми круглыми алмазами.

Сотрудники музея планируют подсчитать долю тех участков окружности, ограничивающей платформу, с которых красный круглый алмаз виден безо всяких помех.

Input

Входной файл содержит несколько (не более 1000) тестовых примеров. Первая строка каждого тестового примера содержит пять целых чисел n, p, x, y, r, где n ($1 \le n \le 100$) — количество оранжевых круглых алмазов, p ($10 \le p \le 1,000$) — радиус фиолетовой платформы, центр которой расположен в начале координат, (x, y) — координаты центра красного алмаза ($-1,000 \le x, y \le 1000$), и r ($0 \le r \le 1000$) — радиус красного алмаза. Гарантируется, что красный алмаз целиком находится на платформе.

В каждой из последующих n строк заданы по три целых числа x_i, y_i и r_i, задающих координаты (x_i, y_i) центра i-го оранжевого алмаза ($-1,000 \le x_i, y_i \le 1000$) и его радиус r_i ($0 < r_i \le 1000$). Гарантируется, что все алмазы целиком находятся на платформе и представляющие их круги не перекрываются.

Входной файл завершается тестовым примером, состоящим из пяти нулей, который обрабатывать не требуется.

Output

Для каждого тестового примера выведите в отдельной строке одно вещественное число с точностью 10^{-4} — долю тех участков границы фиолетовой платформы, с которых красный алмаз виден без помех.

Example

redgem.in	redgem.out
4 10 0 0 1 5 0 2 0 5 2 -5 0 2 0 -5 2 0 0 0 0 0	0.3082

Problem I. Science!

Input file:	`science.in`
Output file:	`science.out`
Time limit:	5 seconds
Memory limit:	256 Mebibytes

Психологи провели некоторый эксперимент, суть которого состояла в следующем: каждому из n человек, участвующих в эксперименте, был дан список "запрещённых" для данного участника чисел. В комнате находятся n стульев, на спинках которых написаны числа. На первом шаге эксперимента участники должны занять места так, чтобы никто не сидел на стуле с запрещённым для него номером. Каждый следующий шаг выглядит следующим образом: по команде экспериментатора участники меняются местами, при этом, во-первых, никто не должен садиться на стул, на котором он сидел ранее, во-вторых, участнику нельзя садиться на стул с "запрещённым" для него номером. Процесс продолжается, пока очередной шаг эксперимента возможен.

Перед началом эксперимента психологов заинтересовал вопрос, какое максимальное количество шагов эксперимента возможно (считать, что участники эксперимента будут действовать оптимально).

Input

Входной файл состоит из нескольких (не более 70) тестовых примеров. Первая строка входного файла содержит одно целое число n ($2 \le n \le 80$) — количество участников эксперимента (и количество задействованных стульев. Каждая из последующих n строк содержит по n символов. Если j-й символ в i-й строке равен '`Y`', то i-й участник может садиться на j-й стул (в противном случае соответствующий символ равен '`N`'). Входной файл завершается тестовым примером с $n = 0$, обрабатывать который не требуется.

Output

Для каждого тестового примера в первой строке выведите число k — максимальное количество шагов эксперимента. Если k не равно 0, далее выведите k строк. Каждая из этих строк содержит n целых чисел, разделённых пробелами. i-е число в j-й строке обозначает номер участника, который на j-м шаге сидит на i-м стуле. Если правильных ответов несколько, выведите любой из них.

Example

science.in	science.out
3 YYY NYY YNY 2 YN YN 0	2 3 1 2 1 2 3 0

Problem J. The Worm in the Apple (Division 1 Only!)

Input file:	worm.in
Output file:	worm.out
Time limit:	13 seconds
Memory limit:	256 mebibytes

Червяк по имени Чарли жил в яблоке и не думал его покидать, но в какой-то момент некто взял яблоко и стал его есть... Поэтому вопрос эмиграции стал для Чарли весьма насущным.

Яблоко задано своей оболочкой в трёхмерном пространстве и является выпуклым многогранником. Также заданы точки в яблоке, в которых в данный момент может находиться Чарли. Для каждой такой точки определите минимальное расстояние, которое Чарли должен проползти, чтобы выбраться на поверхность яблока. Размерами червяка пренебречь.

Input

В первой строке входного файла задано целое число n ($4 \le n \le 1,000$) — количество «опорных точек» яблока. В каждой из последующих n строк задано по три целых числа x, y и z ($-10,000 \le x, y, z \le 10,000$) — координаты точки (x, y, z); при этом никакие четыре точки не лежат в одной плоскости. Яблоко задаётся как выпуклая оболочка всех n точек (x, y, z). Далее следует строка, содержащая одно целое число q ($1 \le q \le 10^5$) — количество точек, в которых может находиться Чарли. В последующих q строках заданы по три целых числа x_i, y_i и z_i ($-10,000 \le x, y, z \le 10,000$), задающие i-ю такую точку. Гарантируется, что ни одна из этих q точек не находится за пределами яблока.

Output

Для каждой возможной точки местоположения Чарли выведите с точностью 10^{-4} одно вещественное число — наименьшее расстояние, которое Чарли должен преодолеть, чтобы выбраться на поверхность яблока.

Example

worm.in	worm.out
6 0 0 0 100 0 0 0 100 0 0 0 100 20 20 20 30 20 10 4 1 1 1 30 30 35 7 8 9 90 2 2	1.0 2.8868 7.0 2.00

Problem K. Combinations (Division 2 Only!)

Input file:	`combinations.in`
Output file:	`combinations.out`
Time limit:	2 seconds
Memory limit:	256 Mebibytes

В теории вероятности и статистике часто возникает подзадача выбора m предметов из набора, состоящего из n различных предметов. Для примера, рассмотрим 5 объектов, занумерованных 0, 1, 2, 3, и 4. Тогда существует следующие 10 способов выбрать 3 объекта из этого множества: 012, 013, 014, 023, 024, 034, 123, 124, 134, 234.

Будем упорядочивать эти подмножества лексикографически. Заметим, что при этом сами множества тоже записываются в лексикографическом порядке, например, множество 014 не может быть записано как 041.

В данной задаче требуется найти i-е M-элементное подмножество N-элементного множества.

Так как в данной задаче N не превосходит 36, для нумерации элементов исходного множества используются цифры и прописные латинские буквы, при этом порядок следования таков: 0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ (то есть при лексикографическом упорядочении 'A' следует после '9').

Input

Входной файл состоит из нескольких тестовых примеров (не более 2000). Первая и единственная строка каждого тестового примера содержит 3 целых числа N, M и I, где N — количество элементов в базовом множестве, M — количество элементов, которые требуется выбрать, I — номер требуемого множества (при лексикографическом упорядочивании). При этом $1 \le N \le 36$, $1 \le M \le N$, $1 \le I \le N!/(M!(N-M)!)$. Входной файл завершается тестовым примером с $N = 0$, который обрабатывать не требуется.

Output

Для каждого тестового примера выведите в отдельной строке i-е множество. Формат вывода указан в примере к задаче — номер тестового примера, двоеточие, пробел, строка из M символов, являющаяся ответом к задаче.

Example

combinations.in	combinations.out
5 3 1 5 3 2 5 3 10 4 2 1 4 2 6 10 5 1 10 5 252 11 11 1 36 2 1 36 2 2 36 2 630 36 36 1 36 18 1 36 18 9075135300 0 0 0	1: 012 2: 013 3: 234 4: 01 5: 23 6: 01234 7: 56789 8: 0123456789A 9: 01 10: 02 11: YZ 12: 0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ 13: 0123456789ABCDEFGH 14: IJKLMNOPQRSTUVWXYZ

Problem L. Find a cycle (Division 2 Only!)

Input file:	cycle.in
Output file:	cycle.out
Time limit:	2 seconds
Memory limit:	256 mebibytes

Рассмотрим следующую операцию R над целыми числами:

1. Берём целое положительное число n.

2. Вычисляем m как наибольшее целое число, квадрат которого не превосходит n.

3. Если $n + m$ чётное, то $R(n) = n + m$. В противном случае $R(n) = n - m$.

Будем последовательно вычислять $R(n)$, $R(R(n))$ и так далее. Например, $R(25)$ равно 30, $R(30)$ равно 25. Получили цикл длины 2, или 2-цикл. А если начать с 24, то получим 13-цикл: 24, 28, 23, 19, 15, 18, 22, 26, 21, 17, 13, 16, 20, 24.

Вам задано T целых положительных чисел. Для каждого из них вычислите, какой цикл оно порождает.

Input

В первой строке входного файла задано целое положительное число T ($1 \le T \le 200$). Далее следуют T целых положительных чисел (по одному числу встроке), каждое из которых не превосходит 10^6.

Output

Для каждого числа во входном файле в отдельной выведите длину порождаемого им цикла в формате, приведённом в примере.

Example

cycle.in	cycle.out
3 25 24 1	25: 2-cycle 24: 13-cycle 1: 2-cycle