Problem A. Количество единиц

Input file:	стандартный ввод
Output file:	стандартный вывод
Time limit:	1 секунда
Memory limit:	256 мегабайт

Вася учится в 11 классе и собирается сдавать ЕГЭ по информатике. К сожалению, ему с трудом даются задания, в которых надо посчитать количество единиц в двоичном представлении значения выражения $2^a + 2^b - 2^c$. Вася выписал несколько подобных заданий и решил их. Теперь он хочет проверить свои решения и просит вас написать для этого программу, которая по заданным a, b и c найдет ответ.

Input

В первой и единственной строке ввода через пробел перечислены три целых числа a, b и c ($1 \le c < b < a \le 10^9$).

Output

Вывести количество единиц в двоичной записи значения выражения $2^a + 2^b - 2^c$.

Example

стандартный ввод	стандартный вывод
3 2 1	2

Problem B. Кратеры

Input file:	стандартный ввод
Output file:	стандартный вывод
Time limit:	1 секунда
Memory limit:	256 мегабайт

Поверхность Луны — усеянная множеством кратеров пустыня. За миллионы лет её случайным образом бомбардировали тысячи метеоритов.

Сейчас к Луне подлетает аппарат, на борту которого находятся три датчика, которые будут сброшены на определённый участок поверхности. Для простоты будем считать этот участок квадратом на плоскости, а кратеры — точками в нём. Из-за возможных помех от космического, солнечного и других излучений, их нужно поместить в кратеры. Спутник будет пролетать мимо Луны и в отведённый временной интервал он должен сбросить датчики, чтобы они упали в нужные кратеры.

К сожалению, в самый последний момент обнаружилась неисправность: из памяти аппарата стёрлись координаты выбранных кратеров. Вам поручено вычислить их заново.

Датчики будут передавать данные об участке лунной поверхности, попадающем в треугольник с вершинами в кратерах, в которых находятся датчики, поэтому необходимо разместить датчики так, чтобы площадь участка была как можно больше.

Вычислите, в каких кратерах нужно расположить датчики.

Input

Первая строка содержит целое число N — количество кратеров ($3 \le N \le 2 \times 10^5$).

В следующих N строках идёт описание координат кратеров — пара целых чисел x_i и y_i.

Гарантируется, что каждая составляющая координат кратеров является случайным числом, выбранным равновероятно из отрезка $[-N, N]$.

Output

Выведите три строки. В каждой строке должны содержаться координаты кратера, в который будет помещён датчик. Если решений несколько, выведите любое.

Example

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

Problem C. MSTrikes back!

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2.5 seconds
Memory limit:	256 mebibytes

Задан связный неориентированный взвешенный граф G. Ваша задача — найти такое подмножество рёбер, что после удаления всех остальных рёбер граф G будет по-прежнему связным. Если таких подмножеств несколько, выберите такое, для которого сумма весов рёбер минимальна.

Так как граф G может быть очень большим, для его генерации будет использован специальный алгоритм:

```
void generate (int N, int seed)
{
int x=seed;
for (int i=2; i<=N; i++)
 {
  x=x * 907 % 2333333;
  int T = x;
  for (j = max(1,i-5); j<= i-1; j++)
   {
    x = x * 907 % 2333333;
    int w = T ^ x;
// добавляем в граф ребро веса w, соединяющее вершины i и j
    add_edge (i, j, w);
   }
  }
return;
}
```

Input

Первая строка входа содержит одно целое число T ($1 \le T \le 50$) — количество тестовых примеров.

Каждый тестовый пример состоит из одной строки, содержащей два целых числа N и S — количество вершин графа и изначальное значение seed ($1 \le N \le 10^7$, $1 \le S \le 2333332$).

Output

Для каждого тестового примера выведите в отдельной строке одно целое число — минимальную сумму весов рёбер.

Example

standard input	standard output
3 3010 2016 3 14159 31415 926	750887251 3382896 9210525875

Problem D. Небоскребы

Input file:	стандартный ввод
Output file:	стандартный вывод
Time limit:	1 секунда
Memory limit:	256 мегабайт

В городе Exo есть ровно N небоскребов, которые расположены друг за другом в один ряд. Если смотреть на них слева направо, то их высоты равны: $a_1, a_2, \ldots, a_N$.

В город вторглось ужасное чудовище, имя которому: «Угурбато». Каждым своим ударом оно разрушает один из небоскребов ударом такой силы, что влево и вправо от него расходится ударная волна.

Рассмотрим левую ударную волну. Пусть чудовище ударило по небоскребу под номером i. Если на пути волны встретился уже разрушенный небоскреб, то волна затухает и дальше не идет. Если волна проходит через целый небоскреб под номером k, то он разрушается только в том случае, если $a_i - a_k \geq |i - k|$ (Если небоскреб разрушится, то волна все равно пойдет дальше). Аналогичные утверждения справедливы и для правой ударной волны.

Чудовище наносит всего M ударов, причем оно совершает новый удар только тогда, когда обе ударные волны от предыдущего удара затухли. Мэр города Exo просит вас посчитать сколько небоскребов было разрушено после каждого удара Угурбато.

Input

Первая строка содержит одно целое число N ($1 \leq N \leq 10^5$) — количество небоскребов.

Вторая строка содержит N целых чисел $a_1, a_2, ..., a_N$, разделённых пробелами — высоты небоскребов, перечисленные слева направо ($1 \leq a_i \leq 10^9$).

Третья строка содержит одно целое число M ($1 \leq M \leq N$) — количество ударов чудовища.

Четвертая строка содержит M различных чисел через пробел: $b_1, b_2, ..., b_M$ — номера небоскребов, которые разрушало чудовище в заданном порядке (небоскребы нумеруются слева направо, начиная с единицы). Гарантируется, что никакой небоскреб под номером $b_i (1 \leq i \leq M)$ не был разрушен ударной волной.

Output

Выходные данные должны содержать M строк, где в i-ой строке содержится единственное целое число — количество небоскребов, которые были разрушены после i-го удара чудовища.

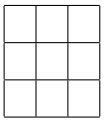
Examples

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

Problem E. Доска

Input file:	стандартный ввод
Output file:	стандартный вывод
Time limit:	1 секунда
Memory limit:	256 мегабайт

Вася учится в первом классе математической школы и делает большие успехи в освоении математики. Сегодня учитель задал детям следующую задачу. Сначала он расчертил доску так, чтобы она представляла собой матрицу $N \times N$, как показано на рисунке (в данном примере $N = 3$).



Далее, учитель продемонстрировал различные варианты, как можно заполнить матрицу числами от 1 до N^2 строго по возрастанию:

(1)

1	2	3
6	5	4
7	8	9

(2)

1	6	7
2	5	8
3	4	9

(3)

1	2	3
8	9	4
7	6	5

(4)

1	8	7
2	9	6
3	4	5

Учитель всегда начинал с верхнего левого угла. В матрице (1) пример представлял собой вертикальную «змейку», в матрице (2) — горизонтальную. Матрицы (3) и (4) представляют собой спираль — в примере (3) заполнение происходит по часовой стрелке, а в примере (4) — против часовой.

Вам необходимо помочь Васе написать программу, которая по заданному N и алгоритму заполнения — (1), (2), (3) или (4) будет печатать результирующую матрицу.

Input

В единственной строке содержатся два натуральных числа N ($1 \le N \le 100$) и a ($1 \le a \le 4$), где a определяет алгоритм заполнения матрицы.

Output

Выведите N строк, состоящих из N чисел, разделенных пробелами, представляющих собой матрицу, заполненную по заданному алгоритму.

Examples

стандартный ввод	стандартный вывод
3 1	1 2 3 6 5 4 7 8 9
3 2	1 6 7 2 5 8 3 4 9
3 3	1 2 3 8 9 4 7 6 5
3 4	1 8 7 2 9 6 3 4 5

Note

В примерах даны входные и выходные данные матриц, предложенных учителем.

Problem F. Числа-друзья

Input file:	стандартный ввод
Output file:	стандартный вывод
Time limit:	1 секунда
Memory limit:	256 мегабайт

У Поликарпа много друзей. Ему это доставляет большое удовольствие, но недавно он задумался, а есть ли друзья у обычных целых чисел?

Поликарп называет два числа *друзьями*, если одно из них делится на другое без остатка.

Например, 2 и 4 будут таковыми, а 10 и 3 нет. В частности, число 1 дружит со всеми.

Поликарпу интересно, можно ли выстроить все целые числа от 1 до n в ряд так, чтобы соседние числа были друзьями? Найдите такую расстановку или сообщите, что её не существует.

Input

В единственной строке содержится целое число n ($1 \le n \le 1000$).

Output

Если ответа не существует, выведите одно число «-1» (без кавычек).

Иначе, выведите n чисел в одной строке через пробел — найденый ряд из n различных чисел.

Если ответов несколько, разрешается вывести любой.

Examples

стандартный ввод	стандартный вывод
2	2 1
3	3 1 2

Problem G. Гмугл

Input file:	стандартный ввод
Output file:	стандартный вывод
Time limit:	1 секунда
Memory limit:	256 мегабайт

В рамках программы импортозамещения Вам поручена разработка нового поисковика под кодовым названием Гмугл (gmoogle). Для первого прототипа Вам требуется показать возможность поиска по базе данных предложений:

- Тексты базы данных соединены в строку S, состоящую из символов 'a'-'z', 'A'-'Z', пробелов, знаков препинания «.!?» (без учета кавычек) а также цифр.
- Символы «.!?» разрывают S на одно или более предложений с одним исключением: если первый символ после точки '.', не являющийся пробелом — буква в нижнем регистре ('a'-'z'), то такая точка предложение не разрывает (пример: «I like tea in a 500 ml. cup» — это одно предложение, а «Cup is 500 ml. I want it» и «Cup is 500 ml. 500 ml is great for me» содержат по два предложения).
- *Словом* называется непрерывная последовательность символов 'a'-'z', 'A'-'Z', ограниченная справа и слева пробелами, знаками препинания или началом/концом строки/предложения. Цифры вплотную к слову находиться не могут, то есть конструкции «10ml» или «R2D2» являются некорректными и в строке S не встретятся.
- В S могут встречаться предложения, не содержащие слов. Гарантируется, что символы «.!?» никогда не находятся рядом друг с другом.

После индексации контента поисковиком пользователи делают поисковые запросы в виде строки q, состоящей из одного или более слов (определение слова дано выше), разделенных пробелами. В начале или конце запроса также могут быть пробелы.

Поисковик должен найти и выдать предложения из базы данных S, где присутствуют все слова из запроса q в любом порядке. Равенством слов считается совпадением всех их букв без учета регистра.

Input

В первой строке входа содержится строка S длиной от 1 до 1000 символов. В следующей строке задано целое число n ($1 \le n \le 100$), — количество поисковых запросов. Далее следуют n строк $q_1, q_2, ..., q_n$, каждая длиной от 1 до 100 символов, в формате, описанном выше. Между, перед и после слов **может** быть любое количество пробелов — как и в строке S.

Output

Для каждого поискового запроса $q_1, q_2, ..., q_n$ поисковик выводит сам запрос на отдельной строке. Далее, каждое в своей строке, выводится список найденных предложений из S, в том порядке, в котором они присутствуют в S. Поисковые запросы и предложения печатаются в кавычках.

В начале и конце предложений пробелы **не печатаются**. См. пример для более точного понимания.

Example

стандартный ввод	стандартный вывод
Hello everyone. I want 2 coffee if you have it. I like coffee very much. 4 HELLO Coffee much coffee VoDka	Search results for "HELLO": - "Hello everyone." Search results for "Coffee": - "I want 2 coffee if you have it." - "I like coffee very much." Search results for "much coffee": - "I like coffee very much." Search results for "VoDka":

Problem H. Генератор

Input file:	`стандартный ввод`
Output file:	`стандартный вывод`
Time limit:	1 секунда
Memory limit:	256 мегабайт

Поликарп и его коллеги празднуют завершение работы над инновационной системой защиты от кибератак. Над системой трудилось множество людей, Поликарп был ответственным за криптографическую часть. Одним из алгоритмов, разработанных им, является поиск случайного простого числа, не превосходящего N. Напомним, что простым числом является число, имеющее ровно два различных делителя — единица и само число.

Алгоритм поиска простого числа, написанный Поликарпом, выглядит так:

1. Выбрать случайное число x из отрезка $[2, N]$. Для всех $N-1$ чисел вероятности быть выбранными одинаковы.
2. Проверить, является ли x простым. Если да, перейти к шагу 3, иначе к шагу 1.
3. Возвратить x как результат.

Проверку того, является ли число x простым, Поликарп реализовал отдельно. Он пользовался тем, что для составного числа всегда существует делитель, не превосходящий квадратного корня из x и при этом отличный от единицы. Это позволяет сэкономить вычисления.

Алгоритм проверки реализован следующим образом:

1. Положить $d := 2$.
2. Если d превосходит квадратный корень из x, то есть $d^2 > x$, то завершить работу алгоритма и сообщить, что x — простое число. Иначе перейти к шагу 3.
3. Проверить, делится ли x на d без остатка. Если да, завершить проверку, сообщив, что x — не простое. Иначе положить $d := d + 1$ и перейти к шагу 2.

Но сегодня Поликарп не мог уснуть. Его беспокоила мысль о том, что его алгоритм поиска может работать вечно! Или хотябы просто долго. А ведь это ставит под угрозу работу их системы.

Самой тяжёлой операцией Поликарп считает деление в проверке на простоту на третьем шаге. Посчитайте математическое ожидание количества этих делений по заданному N.

Input

В первой строке содержится целое число T — количество тестовых примеров ($1 \le T \le 10^5$).

В каждой из следующих T строк содержится один тестовый пример — целое число N ($2 \le N \le 10^7$).

Output

На каждый тест выведите ответ в отдельной строке в виде несократимой дроби.

Example

стандартный ввод	стандартный вывод
6 2 3 4 5 6 10	0/1 0/1 1/2 2/3 1/1 2/1

Problem I. Сложение

Input file:	стандартный ввод
Output file:	стандартный вывод
Time limit:	5 секунд
Memory limit:	256 мегабайт

Это интерактивная задача.

Инженер Поликарп работает в компании «Абак». Каждый день он должен придумывать N двоичных строк длины M. В один из таких дней он решил разнообразить свой рабочий процесс: после того как Поликарп придумывает очередную строку, он интерпретирует её как двоичное число (возможно, с ведущими нулями), поочерёдно складывает его с каждым ранее придуманным числом, и подсчитывает сколько раз результат сложения не поместился в двоичное число длины M.

Поликарп очень плохо складывает числа, поэтому он просит вас помочь ему.

Interaction Protocol

В первой строке программа жюри передает два целых числа N и M ($1 \le N \times M \le 10^5$) — количество придуманных Поликарпом двоичных строк и длину этих строк соответственно.

Далее N раз происходит следующее: программа жюри в отдельной строке передаёт вашей программе очередное придуманное Поликарпом двоичное число, состоящее ровно из M бит, а ваша программа в отдельной строке должна выдать одно целое число — количество раз, когда при сложении данного числа и ранее переданных вам двоичных чисел результат не поместился в двоичное число длины M.

После обработки N запросов ваша программа обязана завершиться с кодом завершения 0. В противном случае результатом взаимодействия будет ошибка (даже если все ответы, выданные вашей программой, были правильными).

Examples

стандартный ввод	стандартный вывод
5 2 01 10 11 00 10	0 0 2 0 2
5 3 110 001 101 110 011	0 0 1 2 3

Note

Для корректной работы программы после каждой операции вывода данных вам необходимо очищать буфер вывода, то есть делать следующие операции:

- В языке Pascal: **flush(output)**;
- В C/C++: **fflush(stdout)** или **cout.flush()**;
- В Java: **System.out.flush()**;
- В Python: **sys.stdout.flush()** из библиотеки **sys**;

- В C#: **Console.Out.Flush**();

Также не забудьте выводить перевод строки после каждого ответа вашей программы.

Problem J. Votter and Paul De Mort

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

Гарри Воттер и Пол Де Морт сражаются в магической дуэли. В какой-то момент Пол Де Морт понял, что его силы заканчиваются, после чего применил заклинание, создающее $N-1$ его копию, так что Гарри теперь противостоят N Полов Де Мортов. Каждые два Пола Де Морта соединены силовой линией в виде отрезка (всего заклинание создало $N(N-1)/2$ силовых линий); каждый экземпляр Пола Де Морта считается точкой.

У Гарри есть мощное заклинание «Магическая гексаграмма», которое строит L-гексаграмму с центром в начале координат. L-гексаграмма представляет собой два равных правильных треугольника, расположенных таким образом, что их пересечение является правильным шестиугольником со стороной L. Значение L является фиксированным; Гарри может менять только угол поворота гексаграммы.

Если Гарри может выбрать угол поворота так, чтобы гексаграмма полностью накрывала все силовые линии, то Пол Де Морт уничтожен. Если же ни при каком угле поворота существует хотя бы одна силовая линия, не накрытая гексаграммой полностью, заклинание не сработает и Гарри должен поменять план действий, выбрав какое-то другое заклинание.

По заданным положениям N Полов Де Мортов и параметру заклинания L выясните, может ли Гарри уничтожить Пола Де Морта с помощью магической гексаграммы или же он должен поменять план действий.

Input

Первая строка входа содержит одно целое число T ($1 \le T \le 1000$) — количество тестовых примеров.

Первая строка каждого тестового примера содержит два целых числа N и L — количество Полов Де Мортов ($3 \le N \le 1000$) и параметр заклинания L ($1 \le L \le 2 \cdot 10^6$).

Далее следуют N строк, каждая из которых содержит два целых числа x_i и y_i, задающих позицию одного экземпляра Пола Де Морта ($-2 \cdot 10^6 \le x_i, y_i \le 2 \cdot 10^6$). Гарантируется, что никакие два экземпляра Пола Де Морта не находятся в одной точке и что сумма всех N во входе не превосходит $3.5 \cdot 10^5$.

Output

Для каждого тестового примера в отдельной строке выведите "`Cast`", если существует такой поворот L-гексаграммы, при котором все силовые поля накрыты, и "`Change`" в противном случае.

Example

standard input	standard output
2 4 10575 2462 9588 -16608 4264 -2462 -9588 16608 -4264 4 12497 -733 17594 -16629 -5795 733 -17594 16629 5795	Cast Change

Problem K. GCD on the segments

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

Задана последовательность целых положительных чисел A длины N..

Требуется выбрать максимальное количество непересекающихся непрерывных подпоследовательностей A так, чтобы все эти подпоследовательности имели одинаковый наибольший общий делитель (наибольшим общим делителем подпоследовательности называется наибольший общий делитель всех её элементов), а также подсчитать, сколькими способами можно сделать такой выбор.

Input

Первая строка входа содержит одно целое число N — количество элементов в последовательности A. Во второй строке заданы N целых чисел A_i — элементы последовательности в порядке возрастания индексов ($1 \le A_i \le 2.5 \cdot 10^6$).

Output

Выведите два целых числа — максимальное количество непересекающихся непрерывных подпоследовательностей последовательности A, обладающих указанным выше свойством, и остаток от деления количества способов выбрать эти последовательности на $10^9 + 7$.

Example

standard input	standard output
3 1 2 5	2 1
6 3 1 3 3 3 3	5 1
3 1 3 1	2 3

Problem L. Уравнение Фибоначчи

Input file:	стандартный ввод
Output file:	стандартный вывод
Time limit:	1 секунда
Memory limit:	256 мегабайт

Бизон Миша взял три последовательных числа Фибоначчи: F_n, F_{n+1} и F_{n+2}, изменил их порядок и подставил в качестве коэффициентов квадратного уравнения:

$$Ax^2 + Bx + C = 0$$

Теперь Миша хочет узнать, сколько существует различных вещественных корней у данного уравнения, и просит вас помочь ему.

Input

В единственной строке содержится три целых неотрицательных числа: i, j и k $(i, j, k \le 10^9)$ — номера членов последовательности Фибоначчи, где $A = F_i$, $B = F_j$ и $C = F_k$. Гарантируется, что все три числа i, j, k различны и что среднее по величине отличается как от большего, так и от меньшего на единицу.

Output

В единственную строку выведите одно целое число — количество различных вещественных корней уравнения.

Examples

стандартный ввод	стандартный вывод
1 2 0	2
1 0 2	0

Note

Последовательность Фибоначчи строится по следующим правилам:

$$F_0 = 0$$

$$F_1 = 1$$

$$F_i = F_{i-1} + F_{i-2}, \text{ где } i > 1$$