A. G-Avoiding Sequence

Given a set of distinct integers S and an integer G, we call a sequence of integers a G-Avoiding Sequence if the following two conditions hold:

1. the sequence is a permutation of the elements of S; and
2. for any two consecutive elements A and B in the sequence, $A - B$ is not divisible by G.

Compute the number of G-Avoiding Sequences modulo the prime 1,234,567,891.

Input

The input consists of several test cases. The first line of each test case contains two integers N ($1 \leq N \leq 200$), the size of S, and G ($1 \leq G \leq 1,000$). The next line will contain N integers, which are the elements of S and between 0 and 10^6. Input is followed by a single line with $N = G = 0$, which should not be processed.

Output

For each test case, output a single line containing the number of G-Avoiding Sequences modulo the prime 1,234,567,891.

Sample Input

```
4 3
1 2 3 4
3 100
10 110 42
0 0
```

Sample Output

```
12
2
```

(Almost empty page.)

B. Lollipop

Byteasar runs a confectionery in Byteburg. Strawberry-vanilla flavoured lollipops are the favourite of local children. These lollipops are composed of multiple segments of the same length, each segment of either strawberry or vanilla flavour. The price of the lollipop is the sum of the values of its segments, where a vanilla segment costs one bythaler, while a strawberry segments costs two.

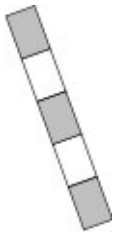

Figure 1: An exemplary lollipop of five segments, three strawberry flavoured and two vanilla, alternately. The price of this lollipop is 8 bythalers.

Currently Byteasar is left with only one lollipop, though possibly very long. As a salesman, he knows only too well that probably no one will want to buy the whole lollipop. For this reason he thinks of breaking the lollipop at the joints of the segments in order to get a shorter lollipop. Each fragment for sale, of course, must stay in one piece.

Byteasar vast experience of a salesman, as well as his understanding of children psychology, tell him that his young customers will most likely want to spend all their money on a single lollipop. With this in mind, he wonders for which values of k the lollipop he has can be broken down in such a way that as a result one would get, among other pieces, a lollipop worth exactly k bythalers. Naturally, he is interested in the way of breaking the lollipop as well. As this task overwhelms him, he asks you for help.

Input

There are multiple test cases in the input file terminated by EOF. For each test case:

In the first line of the standard input there are two integers n and m ($1 \leq n, m \leq 1000000$), separated by a single space. These denote, respectively, the number of segments of the last lollipop left in store, and the number of values of k to consider. The lollipop's segments are numbered from 1 to n, The second line gives an n-letter description of the lollipop, consisting of letters `T` and `W`, where `T` denotes a strawberry flavoured segment, while `W`- vanilla flavoured; the i-th these letters specifies the flavour of the i-th segment. In the following m lines successive values of k ($1 \leq k \leq 2000000$) to consider are given, one per line.

Output

For each test case, your program should print out exactly m lines, giving, one per line, the results for successive values of k, to the standard output. If for a given value of k it is impossible to break the lollipop in such a way that there is a contiguous fragment worth exactly k bythalers, then the word `NIE` (no in Polish) should be printed. Otherwise, two integers l and r ($1 \leq l \leq r \leq n$), separated by single spaces, should be printed, such that the fragment of the lollipop composed of the segments numbered from l to r inclusively is worth exactly k bythalers. If there are multiple such pairs, your program is free to choose one arbitrarily.

Sample Input

```
5 3
TWTWT
5
1
7
5 3
TWTWT
5
1
7
```

Sample Output

```
1 3
2 2
NIE
1 3
2 2
NIE
```

C. Young Hacker

Hackers often have to crack passwords for different data encryption systems. A novice hacker Bill faced such a problem one day. After performing several experiments he noticed regularity in the formation of an encryption key. He knew that a key is an odd integer K, such that K^2 does not divide $(K-1)!$ and its value is in the range $[A, B]$ $(A \leq K \leq B)$. Note, that $(K-1)! = (K-1)(K-2)\cdots 2 \cdot 1)$. He was not able to advance further due to his poor mathematics.

To help young hacker you have to find all possible values of the key.

Input

There are multiple test cases in the input file terminated by EOF. For each test case, input consists of two integers A and B $(3 \leq A < B \leq 10^{18}, B - A \leq 100)$.

Output

For each test case, your program has to print to a single line of output all possible values of key K in ascending order divided by a single space. It is guaranteed that there is at least one key in the range.

Sample Input

```
3 8
```

Sample Output

```
3 5 7
```

(Almost empty page.)

D. Leapers

There is a chessboard of the size $M \times N$. K fairy chess pieces called (p, q)-leapers $(p < q)$ are placed in some squares on this board. Leapers move is similar to a regular chess knights move, with some constraints though. When (p, q)-leaper moves, it can move p squares horizontally and q squares vertically (only upward), or q squares horizontally (only leftward) and p squares vertically. In other words, the move to q squares must be in a direction where corresponding coordinate decreases. Moving outside of the board is prohibited. However several leapers are allowed to occupy the same square. Two players are playing the game, alternating moves. In his turn a player chooses some leaper and moves it according to the rules. The player who is not able to move any leaper loses the game. Given a board configuration determine the winner, assuming both players play optimally.

Input

There are multiple test cases in the input file terminated by EOF. For each test case:

The first line of input contains 5 integers: M, N, K, p, q $(1 \leq M, N \leq 10^9, 1 \leq K \leq 10^5, 1 \leq p < q \leq 20)$. Each of following K lines contains coordinates r_i and c_i of corresponding leaper $(1 \leq r_i \leq M, 1 \leq c_i \leq N)$.

Output

For each test case, the single line of output should contain string `First`, if the first player wins the game under optimal strategy, and `Second` otherwise.

Sample Input

```
10 10 2 1 2
3 7
7 3
7 4 3 1 3
2 3
1 5
4 3
```

Sample Output

```
Second
First
```

(Almost empty page.)

E. The Very Greatest Common Divisor

You need to find greatest common divisor of two integers a and b. Each number a and b are determinants of the square matrix of the form:

$$\begin{pmatrix} 1 & 1 & 0 & \cdots & 0 \\ -1 & 1 & 1 & \ddots & \vdots \\ 0 & -1 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & 1 \\ 0 & \cdots & 0 & -1 & 1 \end{pmatrix}$$

Input

The first line of the input file contains number $n < 250$ of test cases. The description of a test case consists of two lines. The first line contains integer a ($0 < a < 10^{12540}$), the second integer b ($0 < b < 10^{12540}$).

Output

For each test case print the greatest common divisor of integers a and b on a separate line.

Sample Input

```
3
2
3
3
21
6765
610
```

Sample Output

```
1
3
5
```

(Almost empty page.)

F. Contact

Little Petya is preparing for the first contact with aliens. He knows that alien spaceships have shapes of non-degenerate triangles and there will be exactly 4 ships. Landing platform for a ship can be made of 3 special columns located at some points of a Cartesian plane such that these 3 points form a triangle equal to the ship with respect to rotations, translations (parallel shifts along some vector) and reflections (symmetries along the edges). The ships can overlap after the landing.

Each column can be used to land more than one ship, for example, if there are two equal ships, we don't need to build 6 columns to land both ships, 3 will be enough. Petya wants to know what minimum number of columns will be enough to land all ships.

Input

There are multiple test cases in the input file terminated by EOF. For each test case:

Each of 4 lines will contain 6 integers $x_1 y_1 x_2 y_2 x_3 y_3$ ($0 \le x_1, y_1, x_2, y_2, x_3, y_3 \le 20$), representing 3 points that describe the shape of each of 4 ships. It is guaranteed that 3 points in each line will represent a non-degenerate triangle.

Output

For each test case, first line should contain minimum number of columns enough to land all spaceships.

Sample Input

```
0 0 1 0 1 2
0 0 0 2 2 2
0 0 3 0 1 2
0 0 3 0 2 2
0 0 0 1 1 1
0 0 0 2 2 2
0 0 0 5 5 5
0 0 0 17 17 17
```

Sample Output

```
4
9
```

Note

In the first test case columns can be put in these points: $(0, 0), (1, 0), (3, 0), (1, 2)$. Note that the second ship can land using last 3 columns.

In the second test case following points can be chosen: $(0, 0)$, $(0, 1)$, $(1, 0)$, $(0, 2)$, $(2, 0)$, $(0, 5)$, $(5, 0)$, $(0, 17)$, $(17, 0)$. It is impossible to use less than 9 columns.

(Almost empty page.)

G. Step

Mirko and Slavko started taking tap dance lessons. This dance consists mostly of tapping the floor with a special kind of shoe. Since Mirko and Slavko are fast learners, they decided to come up with their own choreography.

Tap dance choreography can be described as a sequence consisting of two letters, 'L' and 'R'. 'L' means that you should tap the floor with your left foot, and R with your right foot. Mirko realised that the most exciting parts of tap dancing are the ones in which you don't use the same leg twice in a row. He defined the value of a choreography as the longest subsequence of consecutive elements that doesn't contain two consecutive 'L's or 'R's. As we all know, designing a choreography can be very challenging, with lots of small changes until it's done. For every alteration that Slavko does, he would like to know the current choreography value. One alteration is changing one 'L' to 'R', and vice versa.

Before any alterations are made, the choreography consists only of letters 'L'.

Input

There are multiple test cases in the input file terminated by EOF. For each test case:

The first line of input contains two integers, the choreography length N ($1 \leq N \leq 200000$), and the number of alterations Q ($1 \leq Q \leq 200000$).

Each of the next Q lines contains an integer specifying the position that Mirko and Slavko are altering, in order of alteration.

Output

For each test case, the output must contain Q integers, one per line – the current values of the choreography after each alteration.

Sample Input

```
6 2
2
4
6 5
4
1
1
2
6
```

Sample Output

```
3
5
3
3
3
5
6
```

(Almost empty page.)

H. Pipes

Mirko, the mad plumber, was hired to construct a water supply network between two locations in a city. The city map can be represented as a $R \times S$ grid. Some cells are not suitable for placing water pipes. Locations Mirko needs to connect are placed directly above the top left cell of the grid, and directly below the bottom right cell.

Each suitable cell Mirko can either leave empty or use it for placing one of the following 6 pipe types:

Find the number of ways that pipes can be placed to connect the two locations with a continuous pipe (water must not be spilled). All placed pipe parts must be in use.

Output the solution modulo 10007.

Input

There are multiple test cases in the input file terminated by EOF. For each test case:

The first line of input contains the integers R and S ($2 \leq R, S \leq 10$), the number of rows and columns of the city grid, respectively. Each of the next R lines contains exactly S characters: '`.`' if the cell is suitable for placing pipes, and '`#`' if not.

Output

For each test case, the first and only line of output must contain the required number of ways modulo 10007.

Sample Input

```
2 3
...
.#.
3 3
...
...
...
```

Sample Output

```
1
12
```

(Almost empty page.)

I. Bottleneck

Farmer John is gathering the cows. His farm contains a network of N ($1 \leq N \leq 100,000$) fields conveniently numbered $1..N$ and connected by $N-1$ unidirectional paths that eventually lead to field 1. The fields and paths form a tree.

Each field $i > 1$ has a single one-way, exiting path to field P_i, and currently contains C_i cows ($1 \leq C_i \leq 1,000,000,000$). In each time unit, no more than M_i ($0 \leq M_i \leq 1,000,000,000$) cows can travel from field i to field P_i ($1 \leq P_i \leq N$) (i.e., only M_i cows can traverse the path).

Farmer John wants all the cows to congregate in field 1 (which has no limit on the number of cows it may have). Rules are as follows:

- Time is considered in discrete units.
- Any given cow might traverse multiple paths in the same time unit. However, no more than M_i total cows can leave field i (i.e., traverse its exit path) in the same time unit.
- Cows never move **away** from field #1.

In other words, every time step, each cow has the choice either to

a) stay in its current field

b) move through one or more fields toward field #1, as long as the bottleneck constraints for each path are not violated

Farmer John wants to know how many cows can arrive in field 1 by certain times. In particular, he has a list of K ($1 \leq K \leq 10,000$) times T_i ($1 \leq T_i \leq 1,000,000,000$), and he wants to know, for each T_i in the list, the maximum number of cows that can arrive at field 1 by T_i if scheduled to optimize this quantity.

Consider an example where the tree is a straight line, and the T_i list contains only $T_1 = 5$, and cows are distibuted as shown:

```
Locn:      1---2---3---4      <-- Pasture ID numbers
C_i:       0   1   12  12     <-- Current number of cows
M_i:           5   8   3      <-- Limits on path traversal; field
                                  1 has no limit since it has no exit
```

The solution is as follows; the goal is to move cows to field 1:

```
Tree:      1---2---3---4
t=0        0   1   12  12     <-- Initial state
t=1        5   4   7   9      <-- field 1 has cows from field 2 and 3
t=2        10  7   2   6
t=3        15  7   0   3
t=4        20  5   0   0
t=5        25  0   0   0
```

Thus, the answer is 25: all 25 cows can arrive at field 1 by time t=5.

Input

There are multiple test cases in the input file terminated by EOF. For each test case:

```
* Line 1: Two space-separated integers: N and K

* Lines 2..N: Line i (not i+1) describes field i with three
        space-separated integers: P_i, C_i, and M_i

* Lines N+1..N+K: Line N+i contains a single integer: T_i
```

Output

For each test case, output K lines, line i contains a single integer that is the maximum number of cows that can arrive at field 1 by time T_i.

Sample Input

```
4 1
1 1 5
2 12 7
3 12 3
5
```

Sample Output

```
25
```

J. Roads and Planes

Farmer John is conducting research for a new milk contract in a new territory. He intends to distribute milk to T ($1 \leq T \leq 25,000$) towns conveniently numbered $1..T$ which are connected by up to R ($1 \leq R \leq 50,000$) roads conveniently numbered $1..R$ and/or P ($1 \leq P \leq 50,000$) airplane flights conveniently numbered $1..P$.

Either road i or plane i connects town A_i ($1 \leq A_i \leq T$) to town B_i ($1 \leq B_i \leq T$) with traversal cost C_i. For roads, $0 \leq C_i \leq 10,000$; however, due to the strange finances of the airlines, the cost for planes can be quite negative ($-10,000 \leq C_i \leq 10,000$).

Roads are bidirectional and can be traversed from A_i to B_i or B_i to A_i for the same cost; the cost of a road must be non-negative.

Planes, however, can only be used in the direction from A_i to B_i specified in the input. In fact, if there is a plane from A_i to B_i it is guaranteed that there is no way to return from B_i to A_i with any sequence of roads and planes due to recent antiterror regulation.

Farmer John is known around the world as the source of the world's finest dairy cows. He has in fact received orders for his cows from every single town. He therefore wants to find the cheapest price for delivery to each town from his distribution center in town S ($1 \leq S \leq T$) or to know that it is not possible if this is the case.

Input

There are multiple test cases in the input file terminated by EOF. For each test case:

```
* Line 1: Four space separated integers: T, R, P, and S

* Lines 2..R+1: Three space separated integers describing a road: A_i,
        B_i and C_i

* Lines R+2..R+P+1: Three space separated integers describing a plane:
        A_i, B_i and C_i
```

Output

For each test case, output T lines, for line i output the minimum cost to get from town S to town i, or "`NO PATH`" if this is not possible.

Sample Input

```
6 3 3 4
1 2 5
3 4 5
5 6 10
3 5 -100
4 6 -100
1 3 -10
```

Sample Output

```
NO PATH
NO PATH
5
0
-95
-100
```