Knight Tour

An `RxC` chess board has some squares filled with water. A knight can move `M` squares horizontally and `N` squares vertically, or `M` squares vertically and `N` squares horizontally in a single step. The knight can jump from `(a,b)` to `(c,d)` if and only if `|a-c|=N` and `|b-d|=M` or `|a-c|=M` and `|b-d|=N`. A knight **cannot** jump to squares filled with water. Of course the knight can not jump out the chess board.

The knight starts at square (1,1) (The intersection of 1st row and 1st column).

A square `(i,j)` is *valid* if it can be reached by a knight from the start square at `(1,1)` with finite number of jumps. For each valid square `(i,j)`, we count the number of valid squares `(x,y)` such that the knight can jump from `(x,y)` to `(i,j)` in one step. If the count is even, we say (i,j) is an *even valid square*, otherwise we say (i,j) is an *odd valid square*.

Please count the number of *even valid squares* and *odd valid squares*.

Input

The first line contains four integer `R`,`C`,`M`,`N`, `1 < R,C <= 100`, `0 <= M,N <= 50`, and `M+N > 0`. The next line contains an integer `W`, `0 <= W < R*C`, which is the number of distinct squares filled with water. Each of the next `W` lines contain a pair of integer `x_i`,`y_i`, `1<=x_i<=R`, `1<=y_i<=C`, and `xi + yi > 2`, indicating that the square `(x_i,y_i)` is filled with water.

Output

Output two integer separated by space: the number of *even* and *odd valid squares*.

Example 1

```
Input:
3 3 2 1
0

Output:
8 0
```

(All the squares except for the center one are *valid*. Each of them can be accessed from two other squares, so they are all *even*.)

Example 2

```
Input:
4 4 1 2
2
4 4
2 2

Output:
4 10
```

Deficit Cycle

After spending your years in college learning programming, you eventually decide that this is not a good career path for you. While you decide what to do next, you take a job as a truck drive. You drive a truck between N cities to earn your living. You live in city 0. The highways have the following properties:

- Highways are one-way only.
- Every highway has two endpoints and they shall not be the same city.
- There is at most one highway in either direction between any pair of cities.
- Starting from your home (city 0), you may travel to any other city using those highways.
- You can earn money by traveling on a given highway (someone will pay you for helping transport goods from one city to another).
- The money you earn could be negative (if the fuel is more expensive than what the trip pays).

A deficit cycle is a driving path starting from one city, going through several highways and getting back to the same city with the negative amount of total earnings for the trip. You may repeatedly lose money if you happen to drive along a deficit cycle again and again. You wonder if such a deficit cycle exists given the highway map of those N cities and decide to write a program that tells you whether such a cycle exists in your map.

Input

The first line of the input contains two integers N (1 <= N <= 1000) and M (0 <= M <= 2000), indicating the number of cities and the number of highways. Each of the following M lines contains three integers u, v and w, indicating there is a highway by which you can travel from city u to city v and you can earn w dollars. (0 <= u,v < N, u != v, -1000 <= w <= 1000)

Output

Print one line containing `possible` or `not possible` to indicate if there exists a deficit cycle. (There is a new line character after the output string.)

Example 1

```
Input:
3 3
0 1 1000
1 2 15
2 1 -42

Output:
possible
```

Example 2

```
Input:
4 4
0 1 10
1 2 20
2 3 30
3 0 -60

Output:
not possible
```

Tune Matching

Betty is trying to learn piano and is experimenting with various sequences of notes to see if she can produce same tune using different note sequences. She wants your help to check if two sequences of notes played by her are equivalent or not. Knowing that you might not be familiar with musical notes, she gives you the following information to help you with your task.

Notes are the basic units of music and each note is associated with a frequency. Two notes are perceived similar if they have a ratio of power of 2 (one is half of the other, one is double of other, etc). Therefore any two notes satisfying such property are considered to be same.

There are 12 basic notes in sequence of increasing frequency. Each of these notes is separated from the previous by the same distance called semitone. 7 of these 12 notes are represented by English alphabets from A to G. The distance between them in terms of semitone is as follows: A-B: 2 B-C: 1 C-D: 2 D-E: 2 E-F: 1 F-G: 2 G-A: 2

Also, two accidentals, sharp (#) and flat (b) can be used to modify the notes. A sharp raises the note a semitone and a flat lowers the note a semitone. A note with an accidental can be represented with a symbol '#' or 'b', e e.g A# or Ab. With this scheme, all 12 notes can be represented.

The below figure illustrates the name of notes as per the rules discussed above.

Now that you have basic information about notes and their similarity, help Betty by writing a program to determine similar tunes or melodies.

For instance the tune "A A D C# C# D E E E F# A D G# A" can be rewritten as "B B E D# D# E Gb Gb Gb G# B E A# B" because distances between the semitones are always equal in both.

Input The first line consists of two integers M and T (1<=M<=10^5, 1<=T<=10^4, T<=M). M and T represents number of notes present in the first and second tune respectively. The next two lines contain M and T notes respectively, indicating the notes of the tunes.

Notes in each line are separated by one space and each one is from English alphabet from A to G and possibly followed by a '#' or 'b' accidental sign.

Output Print a single line containing one character 'S' or 'N' indicating if the two tunes are similar or not.

Example 1

```
Input:
16 4
D G A B C D G G G C D E F# G C C
G G C D

Output:
S
```

Example 2

```
Input:
12 2
C C# D D# E F F# G G# A A# B
C D

Output:
N
```

Shortest Path

Cirno has learned how to solve single-source shortest path problem. She decides to give you a challenge to check if you are as talented as she is.

You are working with a directed graph G with n nodes with weights assigned to edges. The node E is the destination node.

Given an integer B, find the number of nodes that have a path to the destination node and the distance is less than or equal to B.

The distance of a path is the sum of the weights of its edges.

Input

The first line contains 4 integer n, E, B, m. n is the number of nodes, E is the node number for the destination node, B is the limit, and m is the number of edges. 1 <= n <= 100, 1 <= E <= n, 1 <= B <= 10^9, 0 <= m <= n*n

In each of the following m lines, there are 3 integers u,v,w, which means there is an edge from u to v with weight w. 1<=u,v<=n, 0 < w < 10^9

There may be parallel edges or self loops in the input graph.

Output A integer, the number of nodes that have desired paths to E.

Example 1

```
Input:
100 1 123 0

Output:
1
```

There is no edge in the graph, the only node that has a path to E is E itself.

Example 2

```
Input:
5 1 1 4
2 1 1
3 1 1
4 5 1
5 3 1

Output:
3
```