2003 Canadian Computing Competition, Stage 2
Day 2, Question 1

Input file: `perm.in`
Output file: `perm.out`
Source file: `n:\perm\perm.___`

Constrained Permutations

A *permutation* on the numbers $1, 2, \ldots, n$ is a linear ordering of the numbers. For example, there are 6 permutations of the numbers 1,2,3. They are 123, 132, 213, 231, 312 and 321. Another way to think of it is removing n disks numbered 1 to n from a bag (without replacement) and recording the order in which they were drawn out.

Mathematicians (and other smart people) write down that there are $n! = n \cdot (n-1) \cdots 3 \cdot 2 \cdot 1$ permutations of the numbers $1, \ldots, n$. We call this "n factorial."

For this problem, you will be given an integer n ($1 \leq n \leq 9$) and a series of k ($k \geq 0$) constraints on the ordering of the numbers. That is, you will be given k pairs (x, y) indicating that x must come before y in the permutation.

You are to output the number of permutations which satisfy all constraints.

Input

Your input will be $k + 2$ lines. The first line will contain the integer n. The second line will contain the integer k, indicating the number of constraints. The remaining k lines will be pairs of distinct integers which are in the range $1, \ldots, n$.

Output

Your output will be one integer, indicating the number of permutations of $1, \ldots, n$ which satisfy the k constraints.

Sample Input 1

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

Sample Output 1

```
1
```

Sample Input 2

```
4
2
1 2
2 1
```

Sample Output 2

```
0
```

Sample Input 3

```
4
2
1 2
2 3
```

Sample Output 3

```
4
```

2003 Canadian Computing Competition, Stage 2
Day 2, Question 2

Input file: `substr.in`
Output file: `substr.out`
Source file: `n:\substr\substr.___`

Longest Substring

Within a sequence S of integers, find the longest contiguous subsequence that contains every integer at most once. In other words, find the longest contiguous subsequence in which no integer is repeated. If there are several such subsequences, find the one that occurs first in S.

Input

The input file will consist of the elements of S, one per line, in sequence, followed by 0. Each element of S is a positive integer less than 65536. You should not assume anything about the length of S.

Output

The output file should contain the correct subsequence of S, one element per line.

Sample Input

```
1
9
5
3
1
2
8
3
9
0
```

Sample Output

```
9
5
3
1
2
8
```

2003 Canadian Computing Competition, Stage 2
Day 2, Question 3

Input file: `gas.in`
Output file: `gas.out`
Source file: `n:\gas\gas.___`

Cheap Gas

Every day you drive to work through the city. Gas prices have become exorbitant as of late. You've noticed that the gas prices differ throughout the city. Sometimes the cheapest gas is on the other side of the city, but you wonder if it's worth it to drive all the way there just to fill up on cheap gas. You obviously want to spend as little money on gas as possible, but you don't want to run out along the way (your gas tank has a finite size). You wonder if it's possible to calculate the minimum amount of money you need to spend to get to your office each day.

Lucky for you! You live and work in grid city. In grid city there are streets running east-west and avenues running north-south. The streets are sequentially numbered, with the north-most street being 1. The avenues are also similarly numbered such that the west-most avenue is numbered with 1. Residents of the city often refer to their location with a pair of numbers, indicating the street and avenue they're at: $(3, 2)$ implies that they are at the intersection of the 3rd street and 2nd avenue.

After years of "applied experiments" you have discovered something very uncanny about the city: it takes exactly one litre of gas to move from any intersection to any neighbouring intersection (one block north, east, south, or west). It is acceptable to arrive at your office or a gas station with 0 litres of gas in your tank.

When you get to an intersection with a gas station you can choose to fill up with as much or as little gas as you'd like. You don't want to overfill the tank, though, as it would be wasted gas.

Input

Input begins with a number t, the number of test cases.

Each test case begins with a line with four integers m, n, f, and k. m is the number of streets and n is the number of avenues, $(1 \leq m, n \leq 100)$. f is the maximum capacity of your gas tank, in litres. Your starting location is $(1, 1)$ and your office is at (m, n). You start at $(1, 1)$ with a full tank of gas in your car.

Each of the next k lines contains three numbers: a, b, c: a and b are integers, (a, b) being the location of a gas station, and c is the price of gas at that gas station.

Output

For each test case output the minimum amount of money to be spent on gas, rounded to the nearest penny (with two decimal digits), or "Stranded on the shoulder" should it be

impossible to get to your office without running out of gas.

Sample Input

```
2
5 5 6 2
3 3 0.8
4 2 0.5
8 12 4 2
1 2 2
7 11 4.8
```

Sample Output

```
1.00
Stranded on the shoulder
```