ACM International Collegiate Contest
2013 East Central Regional PRACTICE Contest
Grand Valley State University
University of Cincinnati
University of Windsor
Youngstown State University
November 8, 2013

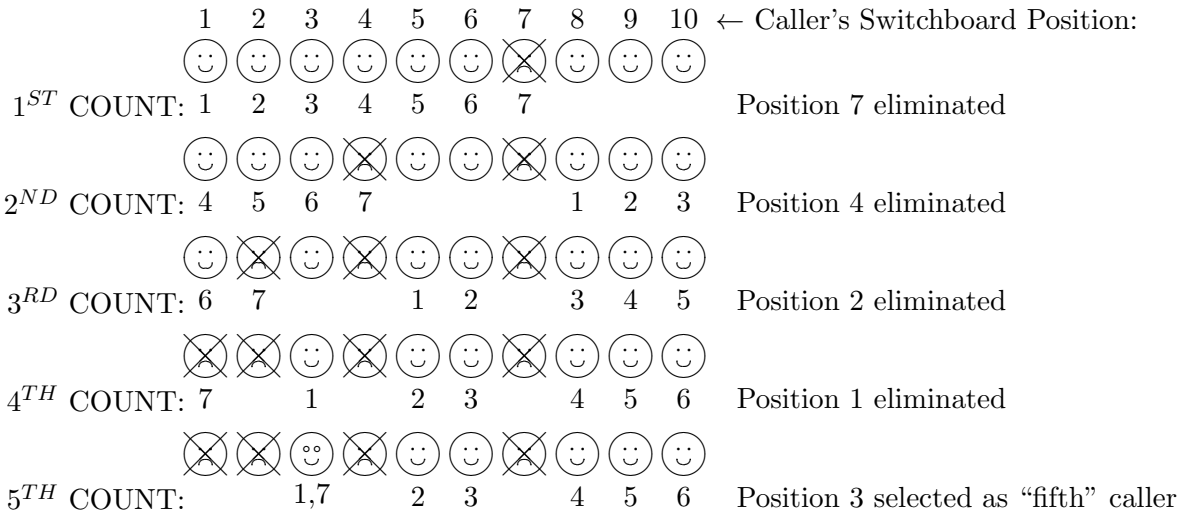
Sponsored by IBM

Rules:

1. There are **three** problems to be completed in **90 minutes**.
2. All questions require you to read the test data from standard input and write results to standard output. You cannot use files for input or output. Additional input and output specifications can be found in the General Information Sheet.
3. When displaying results, follow the format in the Sample Output for each problem. Unless otherwise stated, all whitespace in the Sample Output consists of exactly one blank character.
4. The allowed programming languages are C, C++ and Java.
5. All programs will be re-compiled prior to testing with the judges' data.
6. Non-standard libraries cannot be used in your solutions. The Standard Template Library (STL) and C++ string libraries are allowed. The standard Java API is available, except for those packages that are deemed dangerous by contestant officials (e.g., that might generate a security violation).
7. The input to all problems will consist of multiple test cases.
8. Programming style is not considered in this contest. You are free to code in whatever style you prefer. Documentation is not required.
9. All communication with the judges will be handled by the PC^2 environment.
10. Judges' decisions are to be considered final. No cheating will be tolerated.

Problem A: Calling All Programmers

A local radio station is holding a phone-in contest, and deejay J-Z Phus is in charge of administering the contest. He goes on the air at random times and announces things like "The fifth caller will get a chance for the grand prize." At this point, the switchboard lights up with dozens of callers. All these callers show up on a monitor in front of J-Z numbered 1, 2, 3, etc. J-Z could just pick caller number 5 at this point, but since he figures that everyone basically called in at the same time, he has decided on a different method. He picks a random number – say 7 – and then starts eliminating every seventh caller. If he hits the end of the switchboard while counting, he cycles back to the beginning, but once a caller is eliminated, their position is no longer used in the count. After eliminating 4 callers, he moves down 7 more positions and obtains his "fifth" caller. The figure below show how this would work if J-Z's switchboard held 10 callers. In this case, the caller in position 3 is selected as the "fifth" caller.



Of course, the choice of "fifth" caller, the number of callers, and the use of the number 7 can change from call-in to call-in. You are to write a general program to determine which caller is selected given all of the pertinent information

Input

Each test case will consist of a single line containing 3 positive integers n m k, where n is the number of callers on the switchboard, and m is the number of positions J-Z skips over each time until he gets to the k^{th} caller. The value of n will always be $\geq k$. In the above example, $n = 10, m = 7$ and $k = 5$. The maximum value for any of these values is 200. A line containing three zeros will terminate input.

Output

For each test case, output the position of the caller chosen as the k^{th} caller.

Sample Input

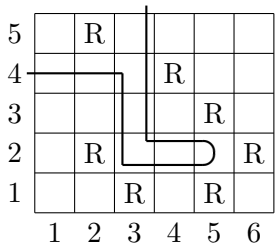
```
10 7 5
20 1 20
0 0 0
```

Sample Output

```
3
20
```

Problem B: Rocky

Sylvester Stallion is an old horse who likes nothing better than to wander around in the fields around his stable. Sylvester is a little single minded and will walk in a straight line unless there is a rock in this path. If that's the case, he does one of three things: 1) if there is no rock to his right, he turns right and continues walking straight in that direction; 2) otherwise, if there is no rock to his left, he turns left and walks in that direction; 3) otherwise, he turns around and walks back the way he came. In a particularly rocky field, he may make several turns in his walk and exit the field in quite an unexpected location. For example, in the field shown below, if Sylvester enters the field at square (1,4), he will follow the path shown (a total of 12 squares), exiting at square (3,5).



Many of his other animal friends are concerned about Sylvester, and would like to know where he ends up on his walks (in particular, his good friend, the ram Beau). That's where you come in – given a description of a field and the location of Sylvester's entrance, you are to determine where he will exit the field and how long it will take him to get there.

Input

Each case starts with three positive integers n m r indicating the number of columns (n) and rows (m) in the field and the number of rocks (r), with $n, m \leq 20$. Following the first line will be lines containing the locations of the r rocks. Each location will be of the form c r, indicating the column and row of the rock. There will be no more than 1 rock in any location. Following the rock locations will be the entrance location for Sylvester. Sylvester's starting direction will always be perpendicular to the side of the field he enters from (this location will never be a corner square) and there will never be a rock in his entrance location square. A line containing three zeros will terminate input.

Output

For each test case, output the case number followed by the last square Sylvester hits before he leaves the field (Sylvester will never get trapped in any field) and the number of squares that Sylvester visited during his walks, counting repeated squares.

Sample Input

```
6 5 7
2 2 2 5 3 1 4 4
5 1 5 3 6 2
1 4
5 5 0
1 2
0 0 0
```

Sample Output

```
Case 1: 3 5 12
Case 2: 5 2 5
```

Problem C: Undercut

The game of Undercut is played between two people who each have five cards numbered 1 through 5. At each turn of the game each player secretly selects one of their cards and then they simultaneously reveal them. Points are then awarded as follows: the player with the higher card gets the number of points on the card *unless* the other player's card is one less; in that case, the player with the lower card gets the sum of the two cards played (this is called an undercut). If both players play the same card, then no points are awarded. For example, if Tessa and Danny are playing Undercut, and Tessa plays a 5 and Danny plays a 3 then Tessa gets 5 points; however if Tessa plays a 5 and Danny plays a 4 instead, Danny would get 5+4=9 points.

Undercut is a great mental game, as each side is thinking "Okay, they think I'll play a 5, so they'll play a 4, so therefore I'll play a 3 to undercut them. But they'll realize I'll think that, so they'll play a 2, so instead I'll just play a 5. But then they'll think..." Unfortunately, we can't model thinking like that in this contest, so instead we'll just give you a list of what each player played and ask you to determine the final score. To make things more interesting, we'll change the rules slightly: if you undercut a 2 with a 1, you'll get 6 points instead of 3 (in the real game, this makes it more worthwhile to play a 1).

Input

The input file will start with an integer n indicating the number of test cases. Each test case will start with a line containing a positive integer $m \leq 100$ indicating the number of rounds for that game. Following this (on one or more subsequent lines) will be m pairs of integers indicating the cards played in each round – the first two numbers will be the cards played by Tessa and Danny, respectively, in the first round, the next two numbers will be their plays in the second round, and so on. Card values will always be between 1 and 5, inclusive.

Output

For each test case, output Tessa's and Danny's scores.

Sample Input

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

Sample Output

```
Game 1: Tessa 5 Danny 14
Game 2: Tessa 16 Danny 26
```