Good Debugging

Any good coder knows that writing a program is only half the battle - and each member of The Team is most certainly a good coder. Maybe you're not a good coder, though, and you're wondering what the other half is. Debugging, of course!

The Team has a program with N ($1 \leq N \leq 10^6$) lines (conveniently numbered $1..N$) - and, unfortunately, every single line happens to initially have a bug in it. The programming language they're using will run the program from the start, line by line, and will always crash as soon as it encounters a total of L ($1 \leq L \leq 10^6$) buggy lines.

The Team will make M ($1 \leq M \leq 10^6$) attempts to debug the program. The ith attempt will consist of modifying every line in the inclusive range of lines $a_i..b_i$ ($1 \leq a \leq b \leq N$). In particular, these coders are so amazing that, every single time they modify a buggy line, it becomes perfect! However, every time they modify a perfect line, they instead introduce a bug into it. After every debugging attempt, The Team will run their new program, and observe how many lines of code it gets through before crashing. Sometimes, the program may even terminate successfully! The modified code will then carry over for future modifications.

Now, the members of The Team would like to know how their program will fare throughout the debugging session. Are your debugging skills good enough to figure that out?

Input

First line: 3 integers, N, M and L

Next M lines: a_i and b_i, for $i=1..N$

Output

M lines: Either 1 integer, the number of lines the program executes before crashing after the first i modifications, or the string "AC?" if it doesn't crash at all, for $i=1..M$.

Example

Input:

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

Output:

```
5
4
AC?
2
```

Explanation of Sample:

The following table shows the status of each line of code after every modification, with newly-changed lines shown in bold font. Buggy lines represented by 1s, and correct ones by 0s.

Modifications	Line 1	Line 2	Line 3	Line 4	Line 5	Line 6
0	1	1	1	1	1	1
1	1	**0**	**0**	**0**	1	1
2	1	0	0	**1**	**0**	**0**
3	**0**	0	0	1	0	0
4	**1**	**1**	0	1	0	0

After 1 modification, then, the program will encounter its second bug at line 5, at which point it will crash. Similarly, after 2 modifications, it will crash after 4 lines, and after 4 modifications, it will crash after just 2. After 3 modifications, however, it will not crash at all, as the program will contain only 1 bug at that point.