The crazy helix

Natural numbers from 1 to N have been placed in an increasing order over some helix (a circular structure). When the helix starts rotating, it is easy to find out

1. The position of a given number
2. The number located at a given position.

The helix has numbers arranged in the following fashion:

```
[1, 2, 3, ..., N]
```

Due to some malfunction, the helix has started rotating in a weird manner. Right now, every possible contiguous interval can be rotated, and hence, locating a particular number or identifying the number at a given position is almost impossible. For example, if at some particular instant, the integer list is like this:

```
[1, 2, 3, 4, 5, ..., N]
```

rotating the interval [5...N] will leave the list like this:

```
[1, 2, 3, 4, N, N - 1, N - 2, ..., 5]
```

We need a software to handle this. Can you help us?

Input Format
The first line of the input consists of two space separated integers, **N, Q**. *N* signifies that initially our list contains numbers from 1 to N, placed in an increasing order. *Q* lines follow and contain input in one of the following formats:

```
1 A B
```

indicating that the helix rotated circularly in the interval [A..B] (both inclusive);

```
2 A
```

indicating that we are interested in knowing the current position of the number A

```
3 A
```

indicating that we are interested in knowing the number at position A.

Output Format
For each line in the input of the form `2 A`

output a line saying

```
element A is at position x
```

where *A* is the number we are interested in and *x* is its current position.

For each line of the form `3 A`

output a line saying

```
element at position A is x
```

where A is the position we are interested in and x is the integer located at this position.

Constraints

$1 \leq N, Q \leq 10^5$
positions are 1-indexed.

Sample Input

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

Sample Output

```
element 3 is at position 1
element at position 3 is 1
element at position 1 is 3
element at position 5 is 1
element 4 is at position 2
element 2 is at position 4
```

Explanation

Initially elements are placed like this:

```
[1, 2, 3, 4, 5]
```

after the first rotation, they are placed like this:

```
[3, 2, 1, 4, 5]
```

and that's how we get the first 2 results (first 2 lines in the output). After second rotation, they are placed like this:

```
[3, 2, 5, 4, 1]
```

and third one does this:

```
[3, 4, 5, 2, 1]
```

In the last rotation (1 5 5), it's easy to see that output matches the initial positions of the elements. Last rotation doesn't change the positions of the elements.