Ivancho and his friends like playing strange games. One of these games is called Rotate-a-palooza.

In this game **N2** of Ivancho’s friends arrange so they form a square with side **N** (so that each of Ivancho’s friends occupies the space of a 1x1 square) and Ivancho instructs them to rotate around in various ways, like by row, by column, or by diagonal. Your task is to write a program, which answers queries of the type „Which one of Ivancho’s friends occupies position **(i, j)** after all rotations are completed?“. Since Ivancho has too many friends and couldn’t possibly remember all their names, he has nicknamed them all with the numbers from **1** to **N2**. As an example, if **22** of Ivancho’s friends participate in the game, their initial arrangement would look like this:

|  |  |
| --- | --- |
| 1 | 2 |
| 3 | 4 |

Ivancho gives his instructions in the following way:

<*rotation\_type*> <*position*> <*number\_of\_rotations*>

Every rotation is given in the following way:

Type of rotation:

1. ‚r’ specifies a rotation of a row
2. ‘c’ specifies a rotation of a column
3. ‘d’ specifies a rotation of a straight diagonal1
4. ‘x’ specifies a rotation of a backward diagonal2

Position is given by:

1. In the cases ‘r’ and ‘c’ – a single positive integer **T**, specifying the row/column, to be rotated (0 ≤ Т ≤ n-1)

2. In the cases ‘d’ and ‘x’ – two positive integers **R** and **C** specifying a diagonal to be rotated (this specifies a single diagonal, which contains the cell **(R,C)** )

Number of rotations is specified:

1. By a single integer **k** (-1000 <= k < 1000)

When one of Ivancho’s friends needs to rotate from а boundary position (i.e. from an (n-1)-th position to the next) we assume that his friend loops back around to the 0-th position again)

For example:

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| 6 | 4 | 5 |
| 7 | 8 | 9 |

r 1 1

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| 6 | 4 | 5 |
| 7 | 8 | 9 |

|  |  |  |
| --- | --- | --- |
| 6 | 2 | 3 |
| 7 | 4 | 5 |
| 1 | 8 | 9 |

c 0 -1

|  |  |  |
| --- | --- | --- |
| 6 | 2 | 3 |
| 8 | 4 | 5 |
| 1 | 7 | 9 |

|  |  |  |
| --- | --- | --- |
| 6 | 2 | 3 |
| 7 | 4 | 5 |
| 1 | 8 | 9 |

d 2 1 -1

|  |  |  |
| --- | --- | --- |
| 6 | 2 | 1 |
| 8 | 3 | 5 |
| 4 | 7 | 9 |

|  |  |  |
| --- | --- | --- |
| 6 | 2 | 3 |
| 8 | 4 | 5 |
| 1 | 7 | 9 |

x 1 1 1

**Input**

The first row of the file rotate.in contains the integer, N – the side of the square, which Ivancho’s friends form.

The next row contains the integer R – the number of rotations, which will be performed.

The next R lines contain the rotations in the described above format.

The next line contains the integer Q – the number of queries of the type „Which one of Ivancho’s friends is at position **(i, j)** after all rotations have been completed?“.

Each of the next Q lines contains two integers i and j

**Output**

In the output file rotate.out print a single positive integer for each query – the number in position **(i, j)** after all rotations have been completed.

**Constraints**

1 ≤ *N* ≤ 1000

1 ≤ R ≤ 104

1 ≤ *Q* ≤ 105

0 ≤ Т ≤ n-1

0 ≤ R, C ≤ n-1

-1000 ≤ *k* ≤ 1000

0 ≤ i, j ≤ n-1

**Time limit: 0.7 sec**

**Memory limit: 256 MB**

**Example test:**

|  |  |
| --- | --- |
| **Input (rotate.in)** | **Output (rotate.out)** |
| 3  4  r 1 1  c 0 -1  d 2 1 -1  x 1 1 1  3  0 0  1 1  2 1 | 6  3  7 |

**Clarifications**

1. A „main diagonal“ of a matrix is the diagonal, containing all cells which lay on the same number row and column (i.e. (0,0) (1,1) … (n-1),(n-1)). A „straight diagonal“ is a diagonal, parallel to the main diagonal. Rotations happen “downward” when k > 0 and “upward” when k < 0, e.g. after d 0 0 1, а00  takes the place of а11, а11 takes the place of а22 etc. After d 0 0 -1 а00 takes the place of аn-1,n-1 etc.

2. A “reverse diagonal” of a matrix is the diagonal, containing all cells with indices (i, n-i-1), e.g. the cells with indices (0,3),(1,2),(2,1),(3,0) comprise the reverse diagonal of a 4x4 matrix. A “backward diagonal” is a diagonal, parallel to the reverse diagonal. Rotations happen “downward”, when k > 0 and “upward”, when k < 0.