Next Cell in Circular Matrix - One Step

|  |  |
| --- | --- |
| **PROBLEM STATEMENT** | Given a 2D Matrix of RxC dimensions.  Given a cell and direction to move in, we need to respond back with indices of next cell after moving one step.  Matrix is treated as circular matrix which means if we are in last column and try to move right, it should take us to the 1st column.  Same we if we are in first row and move UP, we should reach the last row. |
|  |  |
| **INPUT FORMAT** | Line 1:  Integer T - Number of Test Cases. Next T lines follow, each containing R C i j D where R and C are dimensions of matrix i j represent current cell considering matrix is 0-based. D is one of the chars from set ['L', 'R', 'U', 'D'] each representing a direction - LEFT, RIGHT, UP, DOWN |
| **OUTPUT FORMAT** | For each test case print ni nj on a separate line, which are coordinates of next cell after moving one step from i j in given direction., |
| **CONSTRAINTS** | 1 <= T <= 100 1 <= R,C <= 105 0 <= i < R 0 <= j < C D is one of the chars from set ['L', 'R', 'U', 'D'] |

|  |  |  |
| --- | --- | --- |
| **SAMPLE INPUT** | **SAMPLE OUTPUT** | **EXPLANATION** |
| 5 5 5 0 0 R 5 5 0 4 R 3 3 0 0 U 3 3 1 1 L 10 10 8 1 D | 0 1 0 0 2 0 1 0 9 1 | Self Explanatory |

#include <iostream>

#include <vector>

using namespace *std*;

int main()

{

auto testCase = 0, t = 0;

*cin* >> testCase;

t = testCase;

*vector*<*vector*<int>> matrixInfo(testCase);

auto index = 0;

while (t--)

{

auto R = 0, C = 0;

auto i = 0, j = 0;

char direction;

*cin* >> R >> C >> i >> j >> direction;

matrixInfo[index].*push\_back*(R);

matrixInfo[index].*push\_back*(C);

matrixInfo[index].*push\_back*(i);

matrixInfo[index].*push\_back*(j);

matrixInfo[index].*push\_back*(direction);

index++;

}

index = 0;

while (testCase--)

{

auto R = matrixInfo[index].*at*(0);

auto C = matrixInfo[index].*at*(1);

auto i = matrixInfo[index].*at*(2);

auto j = matrixInfo[index].*at*(3);

char direction = matrixInfo[index].*at*(4);

if (direction == 'R')

*cout* << i << " " << (j + 1) % C << *endl*;

else if (direction == 'L')

*cout* << i << " " << (j - 1 + C) % C << *endl*;

else if (direction == 'U')

*cout* << (i - 1 + R) % R << " " << j << *endl*;

else if (direction == 'D')

*cout* << (i + 1) % R << " " << j << *endl*;

index++;

}

return 0;

}