```
/* ------------------------------------------------------------
// maze.h
// ------------------------------------------------------------ */

#ifndef MAZE_H
#define MAZE_H
#include <utility>
#include <vector>
#include <string>
#include <iostream>

using namespace std;

class Maze
{
public:
    // read the maze and store it
    void read_maze(const int height, const int width);

    // print the whole maze
    void print_maze() const;

    // return the maze object by x, y
    char check_maze(const int x, const int y) const;

    // return the robot's origin position
    const pair<int, int> find_robot_in_origin_maze() const;

private:
    vector<vector<char>> body;
};

#endif

/* ------------------------------------------------------------
// maze.cpp
// ------------------------------------------------------------ */

#include "maze.h"

// read the maze and store it
void Maze::read_maze(const int height, const int width)
{
    for(int i = 0 ; i < height ; i++)
    {
        string line;
        getline(cin, line);
        vector<char> tmp;
        for(int j = 0 ; j < width ; j++)
            tmp.push_back(line[j]);
        body.push_back(tmp);
    }
}
```

```
// print the whole maze
void Maze::print_maze() const
{
    for(int i = 0 ; i < body.size() ; i++)
    {
        for(int j = 0 ; j < body[i].size() ; j++)
            cout << body[i][j];
        cout << endl;
    }
}

// return the maze object by x, y
char Maze::check_maze(const int x, const int y) const
{
    if(y < body.size() && x < body[y].size())
        return body[y][x];
    else
        return 0;
}

// return the robot's origin position
const pair<int, int> Maze::find_robot_in_origin_maze() const
{
    for(int i = 0 ; i < body.size() ; i++)
    {
        for(int j = 0 ; j < body[i].size() ; j++)
        {
            if(body[i][j] == 'O')
                return make_pair(j, i);
        }
    }
}

/* ----------------------------------------------------------------
// robot.h
// ---------------------------------------------------------------- */

#ifndef ROBOT_H
#define ROBOT_H
#include <utility>
#include <vector>
#include "maze.h"

using namespace std;

// enumerate four directions
enum Dir
{
    North,
    South,
    West,
    East
};
```

```
class Robot
{
public:
    // initial robot's start position and face direction
    void init(const pair<int, int> in);

    // return robot's position
    const pair<int, int> pos() const;

    // move robot
    void move(const Maze maze);

    // return loop steps
    const long long check_loop();

private:
    // robot's location
    pair<int, int> loc;
    // robot's face direction
    Dir dir = North;
    // robot's journey
    vector<pair<pair<int, int>, Dir>> journey;
    // robot turn right
    void turn();
};

#endif

/* ------------------------------------------------------------
// robot.cpp
// ------------------------------------------------------------ */

#include "robot.h"

// initial robot's start position and face direction
void Robot::init(const pair<int, int> in)
{
    loc = in;
    dir = North;
}

// return robot's position
const pair<int, int> Robot::pos() const
{
    return loc;
}

// move robot
void Robot::move(const Maze maze)
{
    // add journey
    journey.push_back(make_pair(loc, dir));

    int target_x, target_y;
    // turn to right direction
```

```
    while(1)
    {
        //get target x, y
        if(dir == North)
        {
            target_x = loc.first;
            target_y = loc.second - 1;
        }
        else if(dir == East)
        {
            target_x = loc.first + 1;
            target_y = loc.second;
        }
        else if(dir == West)
        {
            target_x = loc.first - 1;
            target_y = loc.second;
        }
        else if(dir == South)
        {
            target_x = loc.first;
            target_y = loc.second + 1;
        }

        if(maze.check_maze(target_x, target_y) == '#')
            turn();
        else
            break;
    }
    loc = make_pair(target_x, target_y);
}

// return loop steps
const long long Robot::check_loop()
{
    for(int i = journey.size() - 1 ; i >= 0 ; i--)
    {
        if(journey[i].first == loc && journey[i].second == dir)
            return journey.size() - i;
    }
    return 0;
}

// robot turn right
void Robot::turn()
{
    if(dir == North)
        dir = East;
    else if(dir == South)
        dir = West;
    else if(dir == West)
        dir = North;
    else
        dir = South;
}
```

TPP2020-HW1-40847011S 高子翔

```
/* ------------------------------------------------------------
// main.cpp
// ------------------------------------------------------------ */

#include <iostream>
#include <string>
#include <vector>
#include <algorithm>
#include <utility>
#include "maze.h"
#include "robot.h"

using namespace std;

int main()
{
    // read width and height
    int width, height;
    cin >> width >> height; cin.ignore();

    // read how many steps
    long long steps;
    cin >> steps; cin.ignore();

    // read maze
    Maze maze;
    maze.read_maze(height, width);

    // init robot
    Robot robot;
    robot.init(maze.find_robot_in_origin_maze());

    // flag for checked the loop or not
    bool loop_flag = false;
    while(steps--)
    {
        robot.move(maze);

        if(!loop_flag)
        {
            long long loop_steps = robot.check_loop();
            if(loop_steps)
            {
                steps %= loop_steps;
                loop_flag = true;
            }
        }
    }

    pair<int, int> answer = robot.pos();
    cout << answer.first << " " << answer.second << endl;
}
```