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

   #ifndef ROBOT_H
   #define ROBOT_H
   #include "maze.h"
   #include <unordered_map>

   class Robot {
      public:
         Robot(int x, int y, long long n, const Maze& maze);

         void execute_moves();

         int get_x() const;

         int get_y() const;

      private:
         int x, y;
         long long n;
         const Maze& maze;
   };

   #endif // ROBOT_H

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


#include "robot.h"
#include <string>
#include <unordered_map>

Robot::Robot(int x, int y, long long n, const Maze& maze)
    : x(x), y(y), n(n), maze(maze) {}

void Robot::execute_moves() {
    int dx[] = {0, 1, 0, -1};
    int dy[] = {-1, 0, 1, 0};
    int direction = 0;
    std::unordered_map<std::string, long long> visited;

    for (long long i = 0; i < n; ++i) {
        std::string state = std::to_string(x) + "," +
std::to_string(y) + "," + std::to_string(direction);
        if (visited.count(state)) {
            int cycle_length = i - visited[state];
            i += ((n - i) / cycle_length) * cycle_length;
        } else {
            visited[state] = i;
        }

        while (true) {
            int nx = x + dx[direction];
            int ny = y + dy[direction];
            if (nx >= 0 && nx < maze.get_width() && ny >= 0 && ny
< maze.get_height() && maze.get_cell(nx, ny) != '#') {
                x = nx;
                y = ny;
                break;
            } else {
                direction = (direction + 1) % 4;
            }
        }
    }
}

int Robot::get_x() const {
    return x;
}

int Robot::get_y() const {
    return y;
}


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



#ifndef MAZE_H
#define MAZE_H

#include <vector>
#include <string>

class Maze {
public:
    Maze(int w, int h, const std::vector<std::string>& map);

    char get_cell(int x, int y) const;

    int get_width() const;

    int get_height() const;

private:
    int width, height;
    std::vector<std::string> grid;
};

#endif // MAZE_H

/* ------------------------------------------------------------
// mzae.cpp
// ---------------------------------------------------------- */

#include "maze.h"

Maze::Maze(int w, int h, const std::vector<std::string>& map)
    : width(w), height(h), grid(map)
```

```
{}

char Maze::get_cell(int x, int y) const {
    return grid[y][x];
}

int Maze::get_width() const {
    return width;
}

int Maze::get_height() const {
    return height;
}

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


#include <iostream>
#include "maze.h"
#include "robot.h"

int main() {
    int w, h;
    long long n;
    std::cin >> w >> h >> n;

    std::vector<std::string> map(h);
    int x, y;
    for (int i = 0; i < h; ++i) {
        std::cin >> map[i];
        size_t pos = map[i].find('O');
        if (pos != std::string::npos) {
            x = pos;
            y = i;
        }
    }

    Maze maze(w, h, map);
    Robot robot(x, y, n, maze);

    robot.execute_moves();

    std::cout << robot.get_x() << " " << robot.get_y() <<
std::endl;

    return 0;
}
```

<< 請適當編排以利列印與閱讀，程式碼儘量不要跨行。 >>