

• main.cpp

```
1 #include <iostream>
2 #include <utility>
3 #include <vector>
4 #include <string>
5 #include "maze.h"
6 #include "robot.h"
7
8 enum Face {
9     up, right, down, left
10 };
11
12 int main() {
13     std::ios_base::sync_with_stdio(false);
14     std::cin.tie(nullptr);
15     unsigned int row, col;
16     unsigned long long step;
17     std::pair<unsigned int, unsigned int> posXY;
18     std::cin >> col >> row >> step;
19     std::vector<std::string> mp;
20     for(size_t i = 0; i < row; ++i) {
21         std::string s;
22         std::cin >> s;
23         for(size_t j = 0; j < col; ++j) {
24             if(s[j] == '0') {
25                 posXY = {i, j};
26                 s[j] = '.';
27                 break;
28             }
29         }
30         mp.push_back(s);
31     }
32     maze mz(row, col, mp);
33     robot bot(posXY, Face::up);
34     bool repeatFlag = false;
35     for(size_t i = 0; i < step; ++i) {
36         std::pair<int, int> nextXY = bot.getNextPos();
37         while(!mz.canWalk(nextXY.first, nextXY.second)) {
38             bot.turn(Face::right);
39             nextXY = bot.getNextPos();
40         }
41         if(!repeatFlag && i > 0) {
42             unsigned long long repeatStep = bot.getRepeatPos();
43             if(repeatStep > 0) {
44                 --repeatStep;
45                 i = step - ((step - repeatStep) % (i - repeatStep)) - 1;
46                 repeatFlag = true;
47                 continue;
48             }
49         }
50         bot.goNext();
51     }
52     posXY = bot.getBotPos();
53     std::cout << posXY.second << " " << posXY.first << std::endl;
54     return 0;
55 }
```

• maze.h

```
1 #pragma once
2 #include <vector>
3 #include <string>
4
5 class maze {
6     private:
7         const unsigned int row, col;
8         const std::vector<std::string> mp;
9     public:
10         maze(const unsigned int, const unsigned int, const std::vector<std::string>);
11         bool canWalk(const int, const int) const;
12 };
```

• maze.cpp

```
1 #include "maze.h"
2
3 maze::maze(const unsigned int row, const unsigned int col, const std::vector<std::string> mp):
4     row(row), col(col), mp(mp) {}
5
6 bool maze::canWalk(const int x, const int y) const {
7     if(x < 0 || x >= row || y < 0 || y >= col || mp[x][y] == '#') return false;
8     return true;
9 }
```

• robot.h

```
1 #pragma once
2 #include <utility>
3 #include <vector>
4 #include <tuple>
5
6 class robot {
7     private:
8         unsigned int x, y, direction;
9         unsigned long long step;
10        std::vector<std::tuple<const unsigned int, const unsigned int, const unsigned int>> history
11        ;
12    public:
13        robot(const std::pair<unsigned int, unsigned int>, const unsigned int);
14        static constexpr int d[4][2] = {{-1, 0}, {0, 1}, {1, 0}, {0, -1}};
15        std::pair<unsigned int, unsigned int> getBotPos() const;
16        std::pair<int, int> getNextPos() const;
17        void turn(const unsigned int);
18        void goNext();
19        unsigned long long getRepeatPos() const;
20 };
```

- robot.cpp

```

1 | #include <iostream>
2 | #include "robot.h"
3 |
4 | constexpr int robot::d[][2];
5 |
6 | robot::robot(const std::pair<unsigned int, unsigned int>posXY, const unsigned int dir) :
7 |     x(posXY.first), y(posXY.second), direction(dir), step(0) {}
8 |
9 | std::pair<unsigned int, unsigned int> robot::getBotPos() const {
10 |     return {x, y};
11 | }
12 |
13 | std::pair<int, int> robot::getNextPos() const {
14 |     return {x + d[direction][0], y + d[direction][1]};
15 | }
16 |
17 | void robot::turn(const unsigned int td) {
18 |     direction = (direction + td) % 4;
19 | }
20 |
21 | void robot::goNext() {
22 |     history.push_back(std::make_tuple(x, y, direction));
23 |     x = x + d[direction][0];
24 |     y = y + d[direction][1];
25 |     ++step;
26 | }
27 |
28 | unsigned long long robot::getRepeatPos() const {
29 |     for(size_t i = 0; i < history.size(); ++i) {
30 |         if(std::get<0>(history[i]) == x && std::get<1>(history[i]) == y && std::get<2>(history[
31 |             i]) == direction) {
32 |             return i + 1;
33 |         }
34 |     }
35 |     return 0;
36 | }

```

- makefile

```

1 | CC = g++
2 | CFLAGS = -O2 -Wall -Wextra -std=c++14
3 | OBJS = main.o maze.o robot.o
4 | EXE = main
5 |
6 | all: $(OBJS)
7 |     $(CC) -o $(EXE) $(OBJS) $(CFLAGS)
8 |
9 | %.o: %.c
10 |
11 | .PHONY: clean
12 | clean:
13 |     ${RM} -r $(OBJS) $(EXE)

```