查核規則	查核結果
Avoid Long Functions & Deep Nesting.	■ 第一版程式碼已遵守規則，未修改
	□ 已修改，簡要說明如下：
Avoid Magic Numbers.	□ 第一版程式碼已遵守規則，未修改
	■ 已修改，簡要說明如下： 賦予地圖上特殊字元（例如:#）名稱
Declare Variables as Locally as Possible.	□ 第一版程式碼已遵守規則，未修改
	■ 已修改，簡要說明如下： 將 enum face 放進 robot class 中
Minimize Global & Shared Data.	■ 第一版程式碼已遵守規則，未修改
	□ 已修改，簡要說明如下：
Always Initialize Variables.	■ 第一版程式碼已遵守規則，未修改
	□ 已修改，簡要說明如下：
Avoid Macros.	■ 第一版程式碼已遵守規則，未修改
	□ 已修改，簡要說明如下：
Use const Proactively.	■ 第一版程式碼已遵守規則，未修改
	□ 已修改，簡要說明如下：
Take Parameters Appropriately by Value, Pointer, or Reference.	■ 第一版程式碼已遵守規則，未修改
	□ 已修改，簡要說明如下：
Hide Information.	■ 第一版程式碼已遵守規則，未修改
	□ 已修改，簡要說明如下：
Know When and How to Code for Scalability.	■ 第一版程式碼已遵守規則，未修改
	□ 已修改，簡要說明如下：
Don’t Optimize Prematurely.	■ 第一版程式碼已遵守規則，未修改
	□ 已修改，簡要說明如下：
Don’t Pessimize Prematurely.	■ 第一版程式碼已遵守規則，未修改
	□ 已修改，簡要說明如下：

- main.cpp

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

int main() {
    std::ios_base::sync_with_stdio(false);
    std::cin.tie(nullptr);
    unsigned int row, col;
    unsigned long long step;
    std::pair<unsigned int, unsigned int> posXY;
    std::cin >> col >> row >> step;
    std::vector<std::string> mp;
    for(size_t i = 0; i < row; ++i) {
        std::string s;
        std::cin >> s;
        if(auto findPos = s.find(maze::robotChar); findPos != std::string::npos) {
            posXY = {i, findPos};
            s[findPos] = maze::freeSpaceChar;
        }
        mp.push_back(s);
    }
    maze mz(row, col, mp);
    robot bot(posXY, robot::face::up);
    bool repeatFlag = false;
    for(size_t i = 0; i < step; ++i) {
        std::pair<int, int> nextXY = bot.getNextPos();
        while(!mz.canWalk(nextXY.first, nextXY.second)) {
            bot.turn(robot::face::right);
            nextXY = bot.getNextPos();
        }
        if(!repeatFlag && i > 0) {
            unsigned long long repeatStep = bot.getRepeatPos();
            if(repeatStep > 0) {
                --repeatStep;
                i = step - ((step - repeatStep) % (i - repeatStep)) - 1;
                repeatFlag = true;
                continue;
            }
        }
        bot.goNext();
    }
    posXY = bot.getBotPos();
    std::cout << posXY.second << " " << posXY.first << std::endl;
    return 0;
}
```

- maze.h

```cpp
#ifndef MAZE_H
#define MAZE_H
#include <vector>
#include <string>

class maze {
  private:
    const unsigned int row, col;
    const std::vector<std::string> mp;
  public:
    maze(const unsigned int, const unsigned int, const std::vector<std::string>);
    static constexpr char obstacleChar = '#', robotChar = 'O', freeSpaceChar = '.';
    bool canWalk(const int, const int) const;
};

#endif
```

- maze.cpp

```cpp
#include "maze.h"

maze::maze(const unsigned int row, const unsigned int col, const std::vector<std::string> mp):
    row(row), col(col), mp(mp) {}

bool maze::canWalk(const int x, const int y) const {
    if(x < 0 || x >= row || y < 0 || y >= col || mp[x][y] == obstacleChar)
        return false;
    return true;
}
```

- robot.h

```cpp
#ifndef ROBOT_H
#define ROBOT_H
#include <utility>
#include <vector>
#include <tuple>

class robot {
  private:
    unsigned int x, y, direction;
    unsigned long long step;
    std::vector<std::tuple<const unsigned int, const unsigned int, const unsigned int>> history
        ;
  public:
    robot(const std::pair<unsigned int, unsigned int>, const unsigned int);
    static constexpr int d[4][2] = {{-1, 0}, {0, 1}, {1, 0}, {0, -1}};
    enum face {up, right, down, left};
    std::pair<unsigned int, unsigned int> getBotPos() const;
    std::pair<int, int> getNextPos() const;
    void turn(const unsigned int);
    void goNext();
    unsigned long long getRepeatPos() const;
};

#endif
```

- robot.cpp

```cpp
#include <iostream>
#include "robot.h"

constexpr int robot::d[][2];

robot::robot(const std::pair<unsigned int, unsigned int>posXY, const unsigned int dir) :
    x(posXY.first), y(posXY.second), direction(dir), step(0) {}

std::pair<unsigned int, unsigned int> robot::getBotPos() const {
    return {x, y};
}

std::pair<int, int> robot::getNextPos() const {
    return {x + d[direction][0], y + d[direction][1]};
}

void robot::turn(const unsigned int td) {
    direction = (direction + td) % 4;
}

void robot::goNext() {
    history.push_back(std::make_tuple(x, y, direction));
    x = x + d[direction][0];
    y = y + d[direction][1];
    ++step;
}

unsigned long long robot::getRepeatPos() const {
    for(size_t i = 0; i < history.size(); ++i) {
        if(std::get<0>(history[i]) == x && std::get<1>(history[i]) == y && std::get<2>(history[
            i]) == direction) {
            return i + 1;
        }
    }
    return 0;
}
```

- makefile

```makefile
CC = g++
CFLAGS = -O2 -Wall -Wextra -std=c++17
OBJS = main.o maze.o robot.o
EXE = main

all: $(OBJS)
	$(CC) -o $(EXE) $(OBJS) $(CFLAGS)

%.o: %.cpp
	$(CC) -c $^ -o $@ $(CFLAGS)

.PHONY: clean
clean:
	${RM} -r $(OBJS) $(EXE)
```