## HW1 修正說明（請參照 Coding Rules 單元投影片，尤其是標題綠字的投影片範例。）

| 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<br>■ 已修改，簡要說明如下：將原先在 main 中裸露的實作過程包裝為新類別 controller 底下的成員函式；修改大部分函式名稱；並簡化其中 runRobot 這個成員函式 |
| Avoid Magic Numbers. | ☐ 第一版程式碼已遵守規則，未修改<br>■ 已修改，簡要說明如下：將定義東南西北方向的數字定義為 robot 底下的成員常數 |
| Declare Variables as Locally as Possible. | ☐ 第一版程式碼已遵守規則，未修改<br>■ 已修改，簡要說明如下：將 runRobot 下變數 loopStepCount 放入回圈內 |
| Minimize Global & Shared Data. | ■ 第一版程式碼已遵守規則，未修改<br>☐ 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<br>■ 已修改，簡要說明如下：在 Maze 和 Robot 兩個類別的變數創建時初始化 |
| Avoid Macros. | ☐ 第一版程式碼已遵守規則，未修改<br>■ 已修改，簡要說明如下：將代表起點、障礙物等數值用全域靜態常數而非巨集 |
| Use const Proactively. | ☐ 第一版程式碼已遵守規則，未修改<br>■ 已修改，簡要說明如下：用 const 修飾一些成員函式和變數，但不是亂加 |
| Take Parameters Appropriately by Value, Pointer, or Reference. | ☐ 第一版程式碼已遵守規則，未修改<br>■ 已修改，簡要說明如下：修正類別 controller 的傳入參數類別為 pointer |
| Hide Information. | ■ 第一版程式碼已遵守規則，未修改<br>☐ 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<br>☐ 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<br>■ 已修改，簡要說明如下：在建構子中才初始化 |
| Don't Pessimize Prematurely. | ☐ 第一版程式碼已遵守規則，未修改<br>■ 已修改，簡要說明如下：全改成前綴 |

```
/*  ------------------------------------------------------------
//  maze.h
//  ------------------------------------------------------------  */
#ifndef GROUP_MAZE_INCLUDED
#define GROUP_MAZE_INCLUDED

#include <string>
#include <vector>
using namespace std;
static constexpr int BEGIN = 0;
static constexpr int UNREACHED_PATH = -1;
static constexpr int OBSTACLE = -2;
static constexpr int UNREACHED_DIR = -3;

class Maze{
public:
	Maze();
	void initMapRow(const string line);
	void setSize(const int width, const int height);
	void setMap(const int y, const int x, const int value);
	void setDirection(const int y, const int x, const int value);
	const int getWidth() const;
	const int getHeight() const;
	const int getMap(const int y, const int x) const;
	const int getDirection(const int y, const int x) const;
private:
	vector<vector<int>> map;
	vector<vector<int>> direction;
	int width;
	int height;
};

#endif

/*  ------------------------------------------------------------
//  maze.cpp
//  ------------------------------------------------------------  */
#include "maze.h"

Maze::Maze() {
	width = 0;
	height = 0;
};
void Maze::initMapRow(const string line) {
	vector<int> mapRowi;
	vector<int> dirRowi;
	for (int j = 0; j < width; ++j) {
		if (line[j] == '#') mapRowi.push_back(OBSTACLE);
		else if (line[j] == '.') mapRowi.push_back(UNREACHED_PATH);
		else if (line[j] == 'O') mapRowi.push_back(BEGIN);
		dirRowi.push_back(UNREACHED_DIR);
	}
	map.push_back(mapRowi);
	direction.push_back(dirRowi);
};
void Maze::setSize(int width, int height) {
	this->height = height;
	this->width = width;
};
void Maze::setMap(int y, int x, int value) {
```

```
	map[y][x] = value;
};
void Maze::setDirection(int y, int x, int value) {
	direction[y][x] = value;
};
const int Maze::getWidth() const {
	return width;
};
const int Maze::getHeight() const {
	return height;
};
const int Maze::getMap(int y, int x) const {
	return map[y][x];
};
const int Maze::getDirection(int y, int x) const {
	return direction[y][x];
};

/*  ------------------------------------------------------------
//  robot.h
//  ------------------------------------------------------------  */
#ifndef GROUP_ROBOT_INCLUDED
#define GROUP_ROBOT_INCLUDED

class Robot {
public:
	Robot();
	void predictNextPos();
	void turnRight();
	void setPosition(const int x, const int y);
	void setNextPosition(const int x, const int y);
	void setTotalSteps(const long long totalSteps);
	const int getRobotDir() const;
	const int getX() const;
	const int getY() const;
	const int getNextX() const;
	const int getNextY() const;
	const long long getTotalSteps() const;
private:
	static constexpr int NORTH = 0;
	static constexpr int EAST = 1;
	static constexpr int SOUTH = 2;
	static constexpr int WEST = 3;
	int x;
	int y;
	int nextX;
	int nextY;
	int robotDir;
	long long totalSteps;
};

#endif

/*  ------------------------------------------------------------
//  robot.cpp
//  ------------------------------------------------------------  */
#include "robot.h"
#include <iostream>

Robot::Robot(){
```

```
	x = 0;
	y = 0;
	nextX = 0;
	nextY = 0;
	robotDir = NORTH;
	totalSteps = 0;
}
ostream & operator << (ostream &os, Robot robot) {
	os << robot.getX() << " " << robot.getY();
	return os;
}
void Robot::predictNextPos() {
	switch(robotDir){
		case NORTH:
			setNextPosition(x, y - 1);
			break;
		case EAST:
			setNextPosition(x + 1, y);
			break;
		case SOUTH:
			setNextPosition(x, y + 1);
			break;
		case WEST:
			setNextPosition(x - 1, y);
			break;
	}
}
void Robot::turnRight() {
	robotDir = (robotDir + 1) % 4;
	predictNextPos();
}
void Robot::setPosition(const int x, const int y) {
	this->x = x;
	this->y = y;
}
void Robot::setNextPosition(const int nextX, const int nextY) {
	this->nextX = nextX;
	this->nextY = nextY;
}
void Robot::setTotalSteps(const long long totalSteps) {
	this->totalSteps = totalSteps;
}
const int Robot::getRobotDir() const {
	return robotDir;
}
const int Robot::getX() const {
	return x;
}
const int Robot::getY() const {
	return y;
}
const int Robot::getNextX() const {
	return nextX;
}
const int Robot::getNextY() const {
	return nextY;
}
const long long Robot::getTotalSteps() const {
	return totalSteps;
}
```

```
/*  ------------------------------------------------------------
//  controller.h
//  ------------------------------------------------------------  */
#ifndef GROUP_CONTROLLER_INCLUDED
#define GROUP_CONTROLLER_INCLUDED

class Robot;
class Maze;
class Controller {
public:
	Controller(Maze *maze, Robot *robot);
	const int getNextCondition() const;
	const int getCurrentCondition() const;
	const int getNextDir() const;
	void setCurrentDir();
	void setNextStep(const int value);
	void input();
	void runRobot();
private:
	Robot *robot;
	Maze *maze;
};

#endif

/*  ------------------------------------------------------------
//  controller.cpp
//  ------------------------------------------------------------  */
#include "robot.h"
#include "maze.h"
#include "controller.h"
#include <iostream>

Controller::Controller(Maze *maze, Robot *robot) {
	this->maze = maze;
	this->robot = robot;
}
const int Controller::getNextCondition() const {
	return maze->getMap(robot->getNextY(), robot->getNextX());
}
const int Controller::getCurrentCondition() const {
	return maze->getMap(robot->getY(), robot->getX());
}
const int Controller::getNextDir() const {
	return maze->getDirection(robot->getNextY(), robot->getNextX());
}
void Controller::setCurrentDir() {
	maze->setDirection(robot->getY(), robot->getX(), robot->getRobotDir());
}
void Controller::setNextStep(const int value) {
	maze->setMap(robot->getNextY(), robot->getNextX(), value + 1);
}
void Controller::input() {
	int width = 0;
	int height = 0;
	long long totalSteps = 0;

	cin >> width >> height; cin.ignore();
	cin >> totalSteps; cin.ignore();
	maze->setSize(width, height);
```

```
	robot->setTotalSteps(totalSteps);
	for (int i = 0; i < maze->getHeight(); ++i) {
		string line;
		getline(cin, line);
		maze->initMapRow(line);
		for (int j = 0; j < maze->getWidth(); ++j) {
			if (maze->getMap(i, j) == BEGIN){
				robot->setPosition(i, j);
				maze->setDirection(i, j, 0);
			}
		}
	}
}
void Controller::runRobot(){
	long long remainingSteps = robot->getTotalSteps();
	bool haveLoop = false;

	for (long long i = 0; i < remainingSteps; ++i) {
		setCurrentDir();
		robot->predictNextPos();
		while (getNextCondition() == OBSTACLE) {
			robot->turnRight();
		}
		if ((getNextCondition() == UNREACHED_PATH || getNextCondition() == BEGIN)
&& !haveLoop) {
			setNextStep(i);
		} else {
			if (getNextDir() == robot->getRobotDir() && !haveLoop) {
				int loopStepCount = getCurrentCondition() - getNextCondition() + 1;
				if (loopStepCount != 0) {
					remainingSteps = (remainingSteps - i - loopStepCount) % loopStepCount;
					robot->setTotalSteps((robot->getTotalSteps() - i - loopStepCount) %
loopStepCount);
					i = 0;
					haveLoop = true;
				}
			}
		}
		robot->setPosition(robot->getNextX(), robot->getNextY());
	}
}

/*  ------------------------------------------------------------
//  main.cpp
//  ------------------------------------------------------------  */
#include "maze.cpp"
#include "robot.cpp"
#include "controller.cpp"

int main() {
	Robot robot;
	Maze maze;
	Controller controller(&maze, &robot);

	controller.input();
	controller.runRobot();
	cout << robot << endl;
	return 0;
}
```