**Pos.h**

#ifndef POS\_H

#define POS\_H

class Pos {

friend const bool operator==(Pos l, Pos r);

private:

//0-based coordinates

int x;

int y;

public:

int getX() const { return x; }

int getY() const { return y; }

void setX(int x) {this -> x = x;}

void setY(int y) {this -> y = y;}

Pos(int x, int y);

Pos(){}

};

const bool operator==(Pos l, Pos r) {

return l.getY() == r.getY() && l.getX() == r.getX();

}

Pos::Pos(int x, int y) {

setX(x);

setY(y);

}

#endif

**Snake.h**

#ifndef SNAKE\_H

#define SNAKE\_H

#include <vector>

#include “Pos.h”

#include <ctime> //time

#include <cstdlib> //srand and rand

#include <algorithm> //reverse

using namespace std;

class Snake {

friend void operator+(Snake& snk, Pos newHead);

private:

vector <Pos> body;

int len;

Pos head;

char curDir;

public:

Snake(int n, int m);

const int getLen() const {return len;}

const char getCell(int x, int y);

const char getCurDir() const { return curDir; }

void setCurDir(char dir) {curDir = dir;}

void setHead(Pos newHead) { head = newHead; }

Pos getHead() const {return head;}

Pos getTail() const {return body[0];}

void cutTale() {body.erase(body.begin());}

void moveHead(Pos newPos) {

head = newPos;

body.push\_back(newPos);

}

};

const char Snake::getCell(int x, int y) {

//Star character will be check outside here

if (head == Pos(x, y)) {

return ‘P’;

}

for (int i = 0; i < len; i++) {

if (body[i] == Pos(x, y)) {

return ‘-’;

}

}

return ‘.’;

}

Snake::Snake(int n, int m) { //n and m are height and width of the board

len = 3;

//Random initialization of coordinates

int y = rand() % n;

int x = rand() % (m - 2); //x-coordinate of one end of the Snake’s body (maybe tail, maybe head)

body.clear();

for (int i = 0; i < len; i++) {

body.push\_back(Pos(x + i, y));

}

if (rand() % 2) { //Head on the right

head = Pos(x + len - 1, y);

curDir = ‘R’;

} else { //Head on the left

head = Pos(x, y);

reverse(body.begin(), body.end());

curDir = ‘L’;

}

}

void operator+(Snake& snk, Pos newHead) {

snk.moveHead(newHead);

snk.len++;

}

#endif

**Board.h**

#ifndef BOARD\_H

#define BOARD\_H

#include “Snake.h”

#include “Pos.h”

#include <ctime> //time

#include <cstdlib> //srand and rand

#include <iostream>

#include <map>

#include <string>

#include <vector>

using namespace std;

map <char, char> opp;

map <char, string> dirName;

map <char, Pos> dirAdd;

class Board {

friend ostream& operator<< (ostream& output, Board board);

private:

int h; //Height

int w; //Width

Snake snk;

Pos foodCor; //Food coordinates

public:

bool checkWin() {

return snk.getLen() >= (double(h \* w)) / 2.0;

}

bool move(char dir) {

int newX = snk.getHead().getX() + dirAdd[dir].getX();

int newY = snk.getHead().getY() + dirAdd[dir].getY();

if (newX >= w || newX < 0 || newY >= h || newY < 0) { //We hit off-board

cout << “Game Over! You crossed the border!\n”;

return false;

} else if (snk.getCell(newX, newY) == ‘-’) { //Snake’s body

cout << “Game Over! You crossed the border!\n”;

return false;

} else if (Pos(newY, newX) == foodCor) { //Food is there

snk + Pos(newX, newY);

snk.setCurDir(dir);

setFoodCor();

return true;

} else { //Empty cell

snk.cutTale();

snk.setCurDir(dir);

snk.moveHead(Pos(newX, newY));

return true;

}

}

const char getCurDir() const {

return snk.getCurDir();

}

Board(int n = 10) : snk(n, n) {

h = n;

w = n;

setFoodCor();

//Set mapping directions utilities

opp[‘R’] = ‘L‘;

opp[‘L‘] = ‘R’;

opp[‘U’] = ‘D’;

opp[‘D’] = ‘U’;

dirName[‘U’] = “UP”;

dirName[‘D’] = “DOWN”;

dirName[‘L’] = “LEFT”;

dirName[‘R’] = “RIGHT”;

dirAdd[‘U’] = Pos(0, -1);

dirAdd[‘D’] = Pos(0, 1);

dirAdd[‘R’] = Pos(1, 0);

dirAdd[‘L’] = Pos(-1, 0);

}

void setFoodCor() {

//Food coordinations initialization

int x = rand() % w;

int y = rand() % h;

while(snk.getCell(x, y) != ‘.’) { //Finding a blank cell to place the food

x = rand() % w;

y = rand() % h;

}

foodCor = Pos(y, x);

}

};

ostream& operator<< (ostream& output, Board board) {

for (int i = 0; i < board.h; i++) {

for (int j = 0; j < board.w; j++) {

if (Pos(i, j) == board.foodCor) {

cout << ‘\*’;

} else {

cout << board.snk.getCell(j, i);

}

}

cout << endl;

}

return output;

}

#endif

**main.cpp**

#include “Board.h”

#include <ctime> //time

#include <cstdlib> //srand and rand

#include <iostream>

int main() {

srand(time(NULL));

Board board; //default w = h = 10 plate size

bool gameIsOver = false;

while (!gameIsOver) {

char opt; //input operation

cin >> opt;

switch (opt) {

case ‘1’: { //plate size

int n;

cin >> n;

board = Board(n);

cout << “The plate dimension is set ” << n << ‘\*’ << n << “.\n”;

} break;

case ‘2’: {

cout << board;

} break;

case ‘3’: {

char dir;

cin >> dir;

if (board.getCurDir() == opp[dir]) {

cout << “The movement to ” << dirName[dir] << “ is not possible!\n”;

} else { //not opposite direction

if (!board.move(dir)) {

gameIsOver = true;

}

}

} break;

default: {

cout << “Please enter 1, 2 or 3\n”;

} break;

}

if (board.checkWin()) {

cout << “Congratulation! you have won!\n”;

gameIsOver = true;

}

}

}