**Câu a:**

#include <iostream>

#include <vector>

using namespace std;

int main() {

int n, m;

cout << "Nhap so dong: ";

cin >> n;

cout << "Nhap so cot: ";

cin >> m;

int matrix[n][m];

cout << "Hay nhap ma tran: " <<endl;

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

for(int j=0; j<m; j++){

cin >> matrix[i][j];

}

}

bool passed[n][m] = {false};

int direction\_x[4] = {-1, 1, 0, 0};

int direction\_y[4] = {0, 0, -1, 1};

int x, y; // vi tri ban dau

cout << "Moi ban nhap vi tri bat dau: " <<endl;

cin >> x >> y;

int steps = 0; // so buoc di

vector<int> path; // luu tru cac o da di qua

while (true) {

// kiem tra o hien tai

passed[x][y] = true;

path.push\_back(matrix[x][y]);

steps++;

// kiem tra cac o xung quanh

int max\_value = -1;

int max\_index = -1;

for (int i = 0; i < 4; i++) {

int nx = x + direction\_x[i];

int ny = y + direction\_y[i];

if (nx >= 0 && nx < n && ny >= 0 && ny < m && !passed[nx][ny] && matrix[nx][ny] > max\_value) {

max\_value = matrix[nx][ny];

max\_index = i;

}

}

// neu khong tim thay o moi, dung lai

if (max\_index == -1) {

break;

}

// di chuyen den o moi

x += direction\_x[max\_index];

y += direction\_y[max\_index];

}

// in cac o robot da di qua

cout << "Quang duong robot da di: "<< steps << endl;

cout << "Robot da di qua nhung o: " << endl;

for (int i = 0; i < path.size(); i++) {

cout << path[i] << " ";

}

cout << endl;

//in cac o chua di qua

cout << "Robot da khong di qua nhung o: " <<endl;

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

for (int j = 0; j < m; j++) {

if (!passed[i][j]) {

cout << matrix[i][j] << " ";

}

}

}

cout << endl;

return 0;

}

***Câu b:***

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

int main() {

int n, m;

cout << "Nhap so dong: ";

cin >> n;

cout << "Nhap so cot: ";

cin >> m;

int matrix[n][m];

cout << "Hay nhap ma tran: " <<endl;

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

for(int j=0; j<m; j++){

cin >> matrix[i][j];

}

}

bool passed[n][m] = {false};

bool passed2[n][m] = {false};

int x, y; // vi tri ban dau

int direction\_x[4] = {-1, 1, 0, 0};

int direction\_y[4] = {0, 0, -1, 1};

cout << "Nhap vi tri bat dau cua robot 1: " <<endl;

cin >> x >> y;

int steps = 0; // so buoc di

vector<int> path; // luu tru cac o robot 1

int x2, y2; // vi tri ban dau cua robot 2

cout << "Nhap vi tri bat dau cua robot 2: " <<endl;

cin >> x2 >> y2;

int steps2 = 0; // so buoc di cua robot 2

vector<int> path2; // luu tru cac o da di qua cua robot 2

while (true) {

// kiem tra o hien tai

passed[x][y] = true;

path.push\_back(matrix[x][y]);

steps++;

// kiem tra o hien tai cua robot 2

passed2[x2][y2] = true;

path2.push\_back(matrix[x2][y2]);

steps2++;

// kiem tra cac o xung quanh

int max\_value = -1;

int max\_index = -1;

for (int i = 0; i < 4; i++) {

int nx = x + direction\_x[i];

int ny = y + direction\_y[i];

if (nx >= 0 && nx < n && ny >= 0 && ny < m && !passed[nx][ny] && matrix[nx][ny] > max\_value) {

max\_value = matrix[nx][ny];

max\_index = i;

}

}

// kiem tra cac o xung quanh robot 1 va 2

int max\_value2 = -1;

int max\_index2 = -1;

for (int i = 0; i < 4; i++) {

int nx2 = x2 + direction\_x[i];

int ny2 = y2 + direction\_y[i];

if (nx2 >= 0 && nx2 < n && ny2 >= 0 && ny2 < m && !passed2[nx2][ny2] && matrix[nx2][ny2] > max\_value2) {

max\_value2 = matrix[nx2][ny2];

max\_index2 = i;

}

}

// neu khong tim thay o moi, dung lai

if (max\_index == -1) {

break;

}

// di chuyen den o moi

x += direction\_x[max\_index];

y += direction\_y[max\_index];

// neu khong tim thay o moi, dung lai

if (max\_index2 == -1) {

break;

}

// di chuyen den o moi cua robot 2

x2 += direction\_x[max\_index2];

y2 += direction\_y[max\_index2];

}

// in cac o robot da di qua

cout << "Quang duong robot 1 da di: "<< steps << endl;

cout << "Robot 1 da di qua nhung o: " << endl;

for (int i = 0; i < path.size(); i++) {

cout << path[i] << " ";

}

// in cac o robot 2 da di qua

cout << "\nQuang duong robot 2 da di: "<< steps2 << endl;

cout << "Robot 2 da di qua nhung o: " << endl;

for (int i = 0; i < path2.size(); i++) {

cout << path2[i] << " ";

}

cout << endl;

//Vi tri cat nhau cua hai robot

int count = 0;

cout << "Hai robot cat nhau tai vi tri: "<<endl;

vector<int> intersection;

set\_intersection(path.begin(), path.end(), path2.begin(), path2.end(), back\_inserter(intersection));

for (int i=0; i<intersection.size();i++) {

cout << intersection[i] << " ";

count ++;

}

cout<<endl;

cout << "Hai robot cat nhau tai "<<count<<" vi tri";

return 0;

}

***Câu C***

#include <iostream>

#include <vector>

using namespace std;

void printMaze(vector<vector<int>> maze, int robot1X, int robot1Y, int robot2X, int robot2Y, int score1, int score2) {

for (int i = 0; i < maze.size(); i++) {

for (int j = 0; j < maze[i].size(); j++) {

if (i == robot1X && j == robot1Y) {

cout << "A\t";

} else if (i == robot2X && j == robot2Y) {

cout << "B\t";

} else if (maze[i][j] == -1) {

cout << "O\t";

} else if (maze[i][j] == -2) {

cout << "X\t";

} else {

cout << maze[i][j] << "\t";

}

}

cout << endl;

}

cout << "Score: " << score1 << " - " << score2 << endl;

}

bool isMoveValid(int currentX, int currentY, int newX, int newY, vector<vector<int>> maze, int robot1X, int robot1Y, int robot2X, int robot2Y) {

if (newX < 0 || newX >= maze.size() || newY < 0 || newY >= maze[0].size()) {

// cout << "Robot da ra khoi me cung!" << endl;

return false;

}

if (maze[newX][newY] == -1 || maze[newX][newY] == -2) {

// cout << "Vi tri nay da di qua!" << endl;

return false;

}

if ((newX == robot1X && newY == robot1Y) || (newX == robot2X && newY == robot2Y)) {

// cout << "Robot khong the di chuyen vao o cua robot khac!" << endl;

return false;

}

return true;

}

int main() {

int numRows, numCols;

cout << "Nhap so hang cua me cung: ";

cin >> numRows;

cout << "Nhap so cot cua me cung: ";

cin >> numCols;

vector<vector<int>> maze(numRows, vector<int>(numCols));

cout << "--------------------------------------------------------------"<<endl;

cout << "Huong dan choi game" << endl;

cout << "W: di chuyen len tren" << endl;

cout << "S: di chuyen xuong duoi" << endl;

cout << "A: di chuyen sang trai" << endl;

cout << "D: di chuyen sang phai" << endl;

cout << "--------------------------------------------------------------"<<endl;

// Nhap gia tri cua me cung

int total\_box\_count = 0;

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

for (int j = 0; j < numCols; j++) {

maze[i][j] = total\_box\_count + 1;

total\_box\_count++;

}

}

int robot1X, robot1Y, robot2X, robot2Y;

cout << "Nhap vi tri x cua Robot 1: ";

cin >> robot1X;

cout << "Nhap vi tri y cua Robot 1: ";

cin >> robot1Y;

cout << "Nhap vi tri x cua Robot 2: ";

cin >> robot2X;

cout << "Nhap vi tri x cua Robot 2: ";

cin >> robot2Y;

int score1 = maze[robot1X][robot1Y], score2 = maze[robot2X][robot2Y];

maze[robot1X][robot1Y] = -2;

maze[robot2X][robot2Y] = -1;

printMaze(maze, robot1X, robot1Y, robot2X, robot2Y, score1, score2);

while (true) {

char direction;

bool isDirectionValid1 = false;

int newX, newY;

while (!isDirectionValid1) {

cout << "Nhap huong di chuyen Robot 1 (W/A/S/D): ";

cin >> direction;

if (direction == 'W' || direction == 'w') {

newX = robot1X - 1;

newY = robot1Y;

isDirectionValid1 = true;

} else if (direction == 'A' || direction == 'a') {

newX = robot1X;

newY = robot1Y - 1;

isDirectionValid1 = true;

} else if (direction == 'S' || direction == 's') {

newX = robot1X + 1;

newY = robot1Y;

isDirectionValid1 = true;

} else if (direction == 'D' || direction == 'd') {

newX = robot1X;

newY = robot1Y + 1;

isDirectionValid1 = true;

} else {

cout << "Robot 1 nhap sai huong di chuyen. Hay nhap (W/A/S/D)." << endl;

continue;

}

}

bool canMove1 = false;

if (robot1X > 0 && isMoveValid(robot1X, robot1Y, robot1X - 1, robot1Y, maze, robot1X, robot1Y, robot2X, robot2Y)) {

canMove1 = true;

} else if (robot1Y > 0 && isMoveValid(robot1X, robot1Y, robot1X, robot1Y - 1, maze, robot1X, robot1Y, robot2X, robot2Y)) {

canMove1 = true;

} else if (robot1X < maze.size() - 1 && isMoveValid(robot1X, robot1Y, robot1X + 1, robot1Y, maze, robot1X, robot1Y, robot2X, robot2Y)) {

canMove1 = true;

} else if (robot1Y < maze[0].size() - 1 && isMoveValid(robot1X, robot1Y, robot1X, robot1Y + 1, maze, robot1X, robot1Y, robot2X, robot2Y)) {

canMove1 = true;

}

if (!canMove1) {

cout << "Robot 1 khong the di chuyen!" << endl;

break;

}

if (!isMoveValid(robot1X, robot1Y, newX, newY, maze, robot1X, robot1Y, robot2X, robot2Y)) {

cout << "Robot 1 khong di chuyen vao o Robot 2 hoac ra khoi me cung!" << endl;

continue;

}

score1 = score1 + maze[newX][newY];

if (maze[newX][newY] != -2) {

maze[newX][newY] = -2;

}

robot1X = newX;

robot1Y = newY;

printMaze(maze, robot1X, robot1Y, robot2X, robot2Y, score1, score2);

bool isDirectionValid2 = false;

while(!isDirectionValid2) {

cout << "Nhap huong di chuyen Robot 2 (W/A/S/D): ";

cin >> direction;

if (direction == 'W' || direction == 'w') {

newX = robot2X - 1;

newY = robot2Y;

isDirectionValid2 = true;

} else if (direction == 'A' || direction == 'a') {

newX = robot2X;

newY = robot2Y - 1;

isDirectionValid2 = true;

} else if (direction == 'S' || direction == 's') {

newX = robot2X + 1;

newY = robot2Y;

isDirectionValid2 = true;

} else if (direction == 'D' || direction == 'd') {

newX = robot2X;

newY = robot2Y + 1;

isDirectionValid2 = true;

} else {

cout << "Robot 2 nhap sai huong di chuyen. Hay nhap (W/A/S/D)." << endl;

continue;

}

}

bool canMove2 = false;

if (robot2X > 0 && isMoveValid(robot2X, robot2Y, robot2X - 1, robot2Y, maze, robot1X, robot1Y, robot2X, robot2Y)) {

canMove2 = true;

} else if (robot2Y > 0 && isMoveValid(robot2X, robot2Y, robot2X, robot2Y - 1, maze, robot1X, robot1Y, robot2X, robot2Y)) {

canMove2 = true;

} else if (robot2X < maze.size() - 1 && isMoveValid(robot2X, robot2Y, robot2X + 1, robot2Y, maze, robot1X, robot1Y, robot2X, robot2Y)) {

canMove2 = true;

} else if (robot2Y < maze[0].size() - 1 && isMoveValid(robot2X, robot2Y, robot2X, robot2Y + 1, maze, robot1X, robot1Y, robot2X, robot2Y)) {

canMove2 = true;

}

if (!canMove2) {

cout << "Robot 2 khong the di chuyen!" << endl;

// score1 = total\_box\_count - score2;

break;

}

if (!isMoveValid(robot2X, robot2Y, newX, newY, maze, robot1X, robot1Y, robot2X, robot2Y)) {

cout << "Robot 2 khong di chuyen vao o Robot 1 hoac ra khoi me cung!" << endl;

continue;

}

score2 = score2 + maze[newX][newY];

if (maze[newX][newY] != -1) {

maze[newX][newY] = -1;

}

robot2X = newX;

robot2Y = newY;

printMaze(maze, robot1X, robot1Y, robot2X, robot2Y, score1, score2);

if ((score1 + score2) == (total\_box\_count-2)) {

// ket thuc game

cout << "Hai robot da di chuyen het cac o!" << endl;

break;

}

}

if (score1 > score2) {

cout << "Robot 1 thang cuoc!" << endl;

} else if (score2 > score1) {

cout << "Robot 2 thang cuoc!" << endl;

} else {

cout << "Hoa nhau!" << endl;

}

return 0;

}

Câu C (Update):

#include <iostream>

#include <vector>

#include <fstream>

#include <cstdlib>

#include <ctime>

using namespace std;

struct robot {

int x, y;

int steps;

vector<int> path;

int score;

};

int direction\_x[4] = {-1, 1, 0, 0};

int direction\_y[4] = {0, 0, -1, 1};

void printMaze(vector<vector<int> > maze, int robot1X, int robot1Y, int robot2X, int robot2Y, int score1, int score2) {

for (int i = 0; i < maze.size(); i++) {

for (int j = 0; j < maze[i].size(); j++) {

if (i == robot1X && j == robot1Y) {

cout << "A\t";

} else if (i == robot2X && j == robot2Y) {

cout << "B\t";

} else if (maze[i][j] == -2) {

cout << "O\t";

} else if (maze[i][j] == -1) {

cout << "X\t";

} else {

cout << maze[i][j] << "\t";

}

}

cout << endl;

}

cout << "Score: " << score1 << " - " << score2 << endl;

}

bool isMoveValid(int currentX, int currentY, int newX, int newY, vector<vector<int> > maze, int robot1X, int robot1Y, int robot2X, int robot2Y) {

if (newX < 0 || newX >= maze.size() || newY < 0 || newY >= maze[0].size()) {

// Robot has gone outside the maze!

return false;

}

if (maze[newX][newY] == -1 || maze[newX][newY] == -2) {

// This position has been visited before!

return false;

}

if ((newX == robot1X && newY == robot1Y) || (newX == robot2X && newY == robot2Y)) {

// Robot cannot move into the other robot's cell!

return false;

}

return true;

}

void calculateWinner(robot RobotA, robot RobotB) {

if (RobotA.score > RobotB.score) {

cout << "Robot A thang voi so diem " << RobotA.score << "!" << endl;

} else if (RobotB.score > RobotA.score) {

cout << "Robot B thang voi so diem " << RobotB.score << "!" << endl;

} else {

cout << "Hoa nhau! Ca hai Robot co cung so diem" << RobotA.score << "!" << endl;

}

}

void move(robot& RobotA, robot& RobotB, vector<vector<int> >& maze) {

int currentX, currentY, newX, newY;

// Robot A's turn

currentX = RobotA.x;

currentY = RobotA.y;

// Find the maximum value among the neighboring cells

int maxScore = -1;

int maxIndex = -1;

for (int i = 0; i < 4; i++) {

newX = currentX + direction\_x[i];

newY = currentY + direction\_y[i];

if (isMoveValid(currentX, currentY, newX, newY, maze, RobotA.x, RobotA.y, RobotB.x, RobotB.y)) {

if (maze[newX][newY] > maxScore) {

maxScore = maze[newX][newY];

maxIndex = i;

}

}

}

// Move Robot A to the cell with the maximum score

if (maxIndex != -1) {

newX = currentX + direction\_x[maxIndex];

newY = currentY + direction\_y[maxIndex];

RobotA.x = newX;

RobotA.y = newY;

RobotA.steps++;

RobotA.score += maze[newX][newY];

maze[currentX][currentY] = -1;

maze[newX][newY] = -1;

}

if (maxIndex == -1 || maxIndex == -2) {

cout << "Game Over!"<<endl;

calculateWinner(RobotA, RobotB);

return;

}

// Print the maze after Robot A's move

printMaze(maze, RobotA.x, RobotA.y, RobotB.x, RobotB.y, RobotA.score, RobotB.score);

// Robot B's turn

currentX = RobotB.x;

currentY = RobotB.y;

// Find the maximum value among the neighboring cells

maxScore = -2;

maxIndex = -2;

for (int i = 0; i < 4; i++) {

newX = currentX + direction\_x[i];

newY = currentY + direction\_y[i];

if (isMoveValid(currentX, currentY, newX, newY, maze, RobotA.x, RobotA.y, RobotB.x, RobotB.y)) {

if (maze[newX][newY] > maxScore) {

maxScore = maze[newX][newY];

maxIndex = i;

}

}

}

// Move Robot B to the cell with the maximum score

if (maxIndex != -2) {

newX = currentX + direction\_x[maxIndex];

newY = currentY + direction\_y[maxIndex];

RobotB.x = newX;

RobotB.y = newY;

RobotB.steps++;

RobotB.score += maze[newX][newY];

maze[currentX][currentY] = -2;

maze[newX][newY] = -2;

}

// Print the maze after Robot B's move

printMaze(maze, RobotA.x, RobotA.y, RobotB.x, RobotB.y, RobotA.score, RobotB.score);

// Recursive call for the next round

move(RobotA, RobotB, maze);

}

void guide() {

cout << "--------------------------------------------------------------" << endl;

cout << "Huong dan choi game" << endl;

cout << "W: di chuyen len tren" << endl;

cout << "S: di chuyen xuong duoi" << endl;

cout << "A: di chuyen sang trai" << endl;

cout << "D: di chuyen sang phai" << endl;

cout << "--------------------------------------------------------------" << endl;

}

int main() {

int numRows, numCols;

cout << "Nhap so hang cua me cung: ";

cin >> numRows;

cout << "Nhap so cot cua me cung: ";

cin >> numCols;

cout << "--------------------------------------------------------------" << endl;

srand(time(0));

vector<vector<int> > maze(numRows, vector<int>(numCols));

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

for (int j = 0; j < numCols; j++) {

maze[i][j] = rand() % 100 + 1;

}

}

cout << "Maze: " << endl;

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

for (int j = 0; j < numCols; j++) {

cout << maze[i][j] << "\t";

}

cout << endl;

}

bool passed[numRows][numCols] = {false};

guide();

robot RobotA;

robot RobotB;

RobotA.steps = 0;

RobotB.steps = 0;

cout << "Nhap vi tri bat dau Robot A: ";

cin >> RobotA.x >> RobotA.y;

cout << "Nhap vi tri bat dau Robot B: ";

cin >> RobotB.x >> RobotB.y;

RobotA.score = maze[RobotA.x][RobotA.y];

RobotB.score = maze[RobotB.x][RobotB.y];

maze[RobotA.x][RobotA.y] = -1;

maze[RobotB.x][RobotB.y] = -2;

printMaze(maze, RobotA.x, RobotA.y, RobotB.x, RobotB.y, RobotA.score, RobotB.score);

move(RobotA, RobotB, maze);

printMaze(maze, RobotA.x, RobotA.y, RobotB.x, RobotB.y, RobotA.score, RobotB.score);

return 0;

}

UPDATE COLOR

#include <iostream>

#include <vector>

#include <fstream>

#include <cstdlib>

#include <ctime>

using namespace std;

// ANSI Escape Codes for text color

#define RESET "\033[0m"

#define RED "\033[31m"

#define GREEN "\033[32m"

#define YELLOW "\033[33m"

#define BLUE "\033[34m"

#define MAGENTA "\033[35m"

#define CYAN "\033[36m"

#define WHITE "\033[37m"

struct robot {

int x, y;

int steps;

int score;

vector<int> path;

};

int direction\_x[4] = {-1, 1, 0, 0};

int direction\_y[4] = {0, 0, -1, 1};

void printMaze(vector<vector<int> > maze, int robot1X, int robot1Y, int robot2X, int robot2Y, int score1, int score2) {

for (int i = 0; i < maze.size(); i++) {

for (int j = 0; j < maze[i].size(); j++) {

if (i == robot1X && j == robot1Y) {

cout << WHITE << "A\t" << RESET;

} else if (i == robot2X && j == robot2Y) {

cout << WHITE << "B\t" << RESET;

} else if (maze[i][j] == -2) {

cout << CYAN << "O\t" << RESET;

} else if (maze[i][j] == -1) {

cout << RED << "X\t" << RESET;

} else {

cout << MAGENTA << maze[i][j] << RESET <<"\t";

}

}

cout << endl;

}

cout << GREEN << "Score: " << score1 << " - " << score2 << RESET << endl;

}

bool isMoveValid(int currentX, int currentY, int newX, int newY, vector<vector<int> > maze, int robot1X, int robot1Y, int robot2X, int robot2Y) {

if (newX < 0 || newX >= maze.size() || newY < 0 || newY >= maze[0].size()) {

// Robot has gone outside the maze!

return false;

}

if (maze[newX][newY] == -1 || maze[newX][newY] == -2) {

// This position has been visited before!

return false;

}

if ((newX == robot1X && newY == robot1Y) || (newX == robot2X && newY == robot2Y)) {

// Robot cannot move into the other robot's cell!

return false;

}

return true;

}

void calculateWinner(robot RobotA, robot RobotB) {

if (RobotA.score > RobotB.score) {

cout << RED << "Robot A thang voi so diem " << RESET << WHITE << RobotA.score << "!" << RESET << endl;

} else if (RobotB.score > RobotA.score) {

cout << BLUE << "Robot B thang voi so diem " << RESET << WHITE << RobotB.score << "!" << RESET << endl;

} else {

cout << GREEN << "Hoa nhau! Ca hai Robot co cung so diem" << RobotA.score << "!" << RESET << endl;

}

cout << RED << "\n--------------------------------------" << RESET << endl;

// in cac o robot da di qua

cout << RED << "Quang duong Robot A da di: "<< RobotA.steps << endl;

cout << RED << "Robot A da di qua nhung o: " << RESET << endl;

for (int i = 0; i < RobotA.path.size(); i++) {

cout << YELLOW << RobotA.path[i] << RESET << " ";

}

cout << CYAN << "\n--------------------------------------" << RESET <<endl;

cout << CYAN << "Quang duong Robot B da di: "<< RobotB.steps << RESET << endl;

cout << CYAN << "Robot B da di qua nhung o: " << RESET << endl;

for (int i = 0; i < RobotB.path.size(); i++) {

cout << YELLOW << RobotB.path[i] << RESET << " ";

}

cout << endl;

ofstream outputFile("output.txt");

if (outputFile.is\_open()) {

outputFile << RobotA.steps <<endl;

for (int i = 0; i < RobotA.path.size(); i++) {

outputFile << RobotA.path[i] << " ";

}

outputFile << "\n\n" << RobotB.steps <<endl;

for (int i = 0; i < RobotB.path.size(); i++) {

outputFile << RobotB.path[i] << " ";

}

outputFile.close();

}

}

void move(robot& RobotA, robot& RobotB, vector<vector<int> >& maze, int numRows, int numCols) {

int currentX, currentY, newX, newY;

// Robot A's turn

currentX = RobotA.x;

currentY = RobotA.y;

// Find the maximum value among the neighboring cells

int maxScore = -1;

int maxIndex = -1;

for (int i = 0; i < 4; i++) {

newX = currentX + direction\_x[i];

newY = currentY + direction\_y[i];

if (isMoveValid(currentX, currentY, newX, newY, maze, RobotA.x, RobotA.y, RobotB.x, RobotB.y)) {

if (maze[newX][newY] > maxScore) {

maxScore = maze[newX][newY];

maxIndex = i;

}

}

}

// Move Robot A to the cell with the maximum score

if (maxIndex != -1) {

newX = currentX + direction\_x[maxIndex];

newY = currentY + direction\_y[maxIndex];

RobotA.x = newX;

RobotA.y = newY;

RobotA.steps++;

RobotA.score += maze[newX][newY];

RobotA.path.push\_back(maze[newX][newY]);

maze[currentX][currentY] = -1;

maze[newX][newY] = -1;

}

if (maxIndex == -1 || maxIndex == -2){

cout << YELLOW << "Game Over!"<< RESET <<endl;

calculateWinner(RobotA, RobotB);

return;

}

// Print the maze after Robot A's move

printMaze(maze, RobotA.x, RobotA.y, RobotB.x, RobotB.y, RobotA.score, RobotB.score);

// Robot B's turn

currentX = RobotB.x;

currentY = RobotB.y;

// Find the maximum value among the neighboring cells

maxScore = -2;

maxIndex = -2;

for (int i = 0; i < 4; i++) {

newX = currentX + direction\_x[i];

newY = currentY + direction\_y[i];

if (isMoveValid(currentX, currentY, newX, newY, maze, RobotA.x, RobotA.y, RobotB.x, RobotB.y)) {

if (maze[newX][newY] > maxScore) {

maxScore = maze[newX][newY];

maxIndex = i;

}

}

}

// Move Robot B to the cell with the maximum score

if (maxIndex != -2) {

newX = currentX + direction\_x[maxIndex];

newY = currentY + direction\_y[maxIndex];

RobotB.x = newX;

RobotB.y = newY;

RobotB.steps++;

RobotB.score += maze[newX][newY];

RobotB.path.push\_back(maze[newX][newY]);

maze[currentX][currentY] = -2;

maze[newX][newY] = -2;

}

// Print the maze after Robot B's move

printMaze(maze, RobotA.x, RobotA.y, RobotB.x, RobotB.y, RobotA.score, RobotB.score);

// Recursive call for the next round

move(RobotA, RobotB, maze, numRows, numCols);

}

void guide() {

cout << "--------------------------------------------------------------" << endl;

cout << CYAN << "Huong dan choi game" << RESET << endl;

cout << CYAN << "W: di chuyen len tren" << RESET << endl;

cout << CYAN << "S: di chuyen xuong duoi" << RESET << endl;

cout << CYAN << "A: di chuyen sang trai" << RESET << endl;

cout << CYAN << "D: di chuyen sang phai" << RESET << endl;

cout << "--------------------------------------------------------------" << endl;

}

int main() {

int numRows, numCols;

do {

cout << RED << "Nhap so hang cua me cung: " << RESET;

cin >> numRows;

cout << CYAN << "Nhap so cot cua me cung: " << RESET;

cin >> numCols;

cout << "--------------------------------------------------------------" << endl;

srand(time(0));

vector<vector<int> > maze(numRows, vector<int>(numCols));

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

for (int j = 0; j < numCols; j++) {

(maze[i][j] = rand() % 100 + 1);

}

}

cout << GREEN << "Maze: " << RESET << endl;

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

for(int j=0; j < numCols; j++){

cout << YELLOW << maze[i][j]<< RESET << "\t";

}

cout << endl;

}

ofstream outputFile("input.txt");

if (outputFile.is\_open()) {

outputFile << numRows << " " << numCols << endl;

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

for (int j = 0; j < numCols; j++) {

outputFile << maze[i][j] << "\t";

}

outputFile << endl;

}

outputFile.close();

}

bool passed[numRows][numCols] = {false};

guide();

robot RobotA;

robot RobotB;

RobotA.steps = 0;

RobotB.steps = 0;

cout << RED << "Nhap vi tri bat dau Robot A: " << RESET;

cin >> RobotA.x >> RobotA.y;

RobotA.path.push\_back(maze[RobotA.x][RobotA.y]);

cout << BLUE << "Nhap vi tri bat dau Robot B: " << RESET;

cin >> RobotB.x >> RobotB.y;

RobotB.path.push\_back(maze[RobotB.x][RobotB.y]);

RobotA.score = maze[RobotA.x][RobotA.y];

RobotB.score = maze[RobotB.x][RobotB.y];

maze[RobotA.x][RobotA.y] = -1;

maze[RobotB.x][RobotB.y] = -2;

printMaze(maze, RobotA.x, RobotA.y, RobotB.x, RobotB.y, RobotA.score, RobotB.score);

move(RobotA, RobotB, maze, numRows, numCols);

cout << GREEN <<"--------------------------------------" << RESET << endl;

char h;

cout << GREEN <<"Ban co muon tiep tuc khong? (Y/N): " << RESET;

cin >> h;

if(h == 'N' || h == 'n'){

break;

}

} while(true);

cout << RED << "Cam on da choi game!" << RESET;

return 0;

}

Project Robots

#include <iostream>

#include "console.h"

#include <windows.h>

#include <ctime>

#include <cstdlib>

#include <stdio.h>

#include <vector>

int numRows = 14;

int numCols = 14;

int direction\_x[4] = {-1, 1, 0, 0};

int direction\_y[4] = {0, 0, -1, 1};

using namespace std;

struct robot {

int x, y;

int steps;

int score;

vector<int> path;

};

void gotoxy (int column, int line){

COORD coord;

coord.X = column;

coord.Y = line;

SetConsoleCursorPosition(GetStdHandle(STD\_OUTPUT\_HANDLE),coord);

}

void TextColor (int color){

SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE) , color);

}

void clrscr(){

CONSOLE\_SCREEN\_BUFFER\_INFO csbiInfo;

HANDLE hConsoleOut;

COORD Home = {0,0};

DWORD dummy;

hConsoleOut = GetStdHandle(STD\_OUTPUT\_HANDLE);

GetConsoleScreenBufferInfo(hConsoleOut,&csbiInfo);

FillConsoleOutputCharacter(hConsoleOut,' ',csbiInfo.dwSize.X \* csbiInfo.dwSize.Y,Home,&dummy);

csbiInfo.dwCursorPosition.X = 0;

csbiInfo.dwCursorPosition.Y = 0;

SetConsoleCursorPosition(hConsoleOut,csbiInfo.dwCursorPosition);

}

void hideCursor(){

HANDLE consoleHandle = GetStdHandle(STD\_OUTPUT\_HANDLE);

CONSOLE\_CURSOR\_INFO cursorInfo;

GetConsoleCursorInfo(consoleHandle, &cursorInfo);

cursorInfo.bVisible = false; // Thi?t l?p tr?ng thái con tr? chu?t không hi?n th?

SetConsoleCursorInfo(consoleHandle, &cursorInfo);

}

void khungngoai(){

TextColor(11);

for(int i = 0; i<=78; i++)

{

gotoxy(i, 0); cout << char(205);

Sleep(1);

}

for(int i = 0; i<=78; i++)

{

gotoxy(i, 46); cout << char(205);

Sleep(1);

}

for(int i = 0; i<=45; i++)

{

gotoxy(0, i); cout << char(186);

Sleep(1);

}

for(int i = 0; i<=45; i++)

{

gotoxy(79, i); cout << char(186);

Sleep(1);

}

gotoxy(0, 0); cout << char(201);

gotoxy(0, 46); cout << char(200);

gotoxy(79, 0); cout << char(187);

gotoxy(79, 46); cout << char(188);

TextColor(11);

for(int i = 2; i<=77; i++)

{

gotoxy(i, 1); cout << char(205);

Sleep(1);

}

for(int i = 1; i<=77; i++)

{

gotoxy(i, 45); cout << char(205);

Sleep(1);

}

for(int i = 1; i<=44; i++)

{

gotoxy(1, i); cout << char(186);

Sleep(1);

}

for(int i = 1; i<=44; i++)

{

gotoxy(78, i); cout << char(186);

Sleep(1);

}

gotoxy(1, 1); cout << char(201);

gotoxy(1, 45); cout << char(200);

gotoxy(78, 1); cout << char(187);

gotoxy(78, 45); cout << char(188);

for(int i = 0; i<15; i++)

{

gotoxy(63+i, 4); cout << char(205);

}

for(int i = 0; i<2; i++)

{

gotoxy(63, i+2); cout << char(186);

}

gotoxy(63, 4); cout << char(200);

gotoxy(67, 2); TextColor(12); cout << "Luot Choi";

TextColor(15); gotoxy(68,3); cout << "-- --";

TextColor(13); gotoxy(71, 3); cout << char(2);

TextColor(11);

for(int i = 0; i<15; i++)

{

gotoxy(2+i, 4); cout << char(205);

}

for(int i = 0; i<2; i++)

{

gotoxy(17, i+2); cout << char(186);

}

for(int i = 0; i<43; i++)

{

gotoxy(19+i, 2); cout << "\_";

}

gotoxy(17, 4); cout << char(188);

gotoxy(34, 0); TextColor(160); cout << " GAME CO CARO ";

gotoxy(25,2); TextColor(12); cout << "TAN CU CHAN - CNTT K16D - ICTU";

gotoxy(5, 2); TextColor(12); cout << "So Quan Co";

TextColor(15);

gotoxy(3, 3); cout << " - | - ";

gotoxy(3, 3); TextColor(14); cout << "X";

gotoxy(11, 3); TextColor(14); cout << "O";

gotoxy(42,40);

}

void khoitaoBanCo(const vector<vector<int> >& maze) {

int m = maze.size();

int n = maze[0].size();

TextColor(10);

cout << char(218);

for (int i = 0; i < n - 1; i++) {

cout << char(196) << char(196) << char(196) << char(194);

}

cout << char(196) << char(196) << char(196) << char(191) << endl;

for (int i = 0; i < m - 1; i++) {

cout << char(179);

for (int k = 0; k < n - 1; k++) {

cout << " " << maze[i][k] << " " << char(179);

}

cout << " " << maze[i][n - 1] << " " << char(179) << endl;

cout << char(195);

for (int j = 0; j < n - 1; j++) {

cout << char(196) << char(196) << char(196) << char(197);

}

cout << char(196) << char(196) << char(196) << char(180) << endl;

}

cout << char(179);

for (int k = 0; k < n - 1; k++) {

cout << " " << maze[m - 1][k] << " " << char(179);

}

cout << " " << maze[m - 1][n - 1] << " " << char(179) << endl;

cout << char(192);

for (int i = 0; i < n - 1; i++) {

cout << char(196) << char(196) << char(196) << char(193);

}

cout << char(196) << char(196) << char(196) << char(217);

}

bool isMoveValid(int currentX, int currentY, int newX, int newY, vector<vector<int> > maze, int robot1X, int robot1Y, int robot2X, int robot2Y) {

if (newX < 0 || newX >= maze.size() || newY < 0 || newY >= maze[0].size()) {

// Robot has gone outside the maze!

return false;

}

if (maze[newX][newY] == -1 || maze[newX][newY] == -2) {

// This position has been visited before!

return false;

}

if ((newX == robot1X && newY == robot1Y) || (newX == robot2X && newY == robot2Y)) {

// Robot cannot move into the other robot's cell!

return false;

}

return true;

}

void veBanCo(const vector<vector<int> >& maze, int robot1X, int robot1Y, int robot2X, int robot2Y, int score1, int score2) {

gotoxy(0,0);

TextColor(10);

int m = maze.size();

int n = maze[0].size();

cout << char(218);

for (int i = 0; i < n - 1; i++) {

cout << char(196) << char(196) << char(196) << char(194);

}

cout << char(196) << char(196) << char(196) << char(191) << endl;

for (int i = 0; i < m - 1; i++) {

cout << char(179);

for (int k = 0; k < n - 1; k++) {

if (i == robot1X && k == robot1Y) {

TextColor(7);

cout << " A ";

TextColor(10);

cout << char(179);

} else if (i == robot2X && k == robot2Y) {

TextColor(7);

cout << " B ";

TextColor(10);

cout << char(179);

} else if (maze[i][k] == -2) {

TextColor(1);

cout << " O ";

TextColor(10);

cout << char(179);

} else if (maze[i][k] == -1) {

TextColor(4);

cout << " X ";

TextColor(10);

cout << char(179);

} else {

cout << " ";

TextColor(13);

cout << maze[i][k];

cout << " ";

TextColor(10);

cout << char(179);

}

}

if (i == robot1X && n - 1 == robot1Y) {

TextColor(7);

cout << " A ";

TextColor(10);

cout << char(179);

} else if (i == robot2X && n - 1 == robot2Y) {

TextColor(7);

cout << " B ";

TextColor(10);

cout << char(179);

} else if (maze[i][n-1] == -2) {

TextColor(1);

cout << " O ";

TextColor(10);

cout << char(179);

} else if (maze[i][n-1] == -1) {

TextColor(4);

cout << " X ";

TextColor(10);

cout << char(179);

} else {

cout << " ";

TextColor(13);

cout << maze[i][n - 1];

cout << " ";

TextColor(10);

cout << char(179);

}

cout << endl;

TextColor(10);

cout << char(195);

for (int j = 0; j < n - 1; j++) {

cout << char(196) << char(196) << char(196) << char(197);

}

cout << char(196) << char(196) << char(196) << char(180) << endl;

}

cout << char(179);

for (int k = 0; k < n - 1; k++) {

if (m - 1 == robot1X && k == robot1Y) {

TextColor(7);

cout << " A ";

TextColor(10);

cout << char(179);

} else if (m - 1 == robot2X && k == robot2Y) {

TextColor(7);

cout << " B ";

TextColor(10);

cout << char(179);

} else if (maze[m-1][k] == -2) {

TextColor(1);

cout << " O ";

TextColor(10);

cout << char(179);

} else if (maze[m-1][k] == -1) {

TextColor(4);

cout << " X ";

TextColor(10);

cout << char(179);

}

else {

cout << " ";

TextColor(13);

cout << maze[m - 1][k];

cout << " ";

TextColor(10);

cout << char(179);

}

}

if (m - 1 == robot1X && n - 1 == robot1Y) {

TextColor(7);

cout << " A ";

TextColor(10);

cout << char(179);

} else if (m - 1 == robot2X && n - 1 == robot2Y) {

TextColor(7);

cout << " B ";

TextColor(10);

cout << char(179);

} else if (maze[m-1][n-1] == -2) {

TextColor(1);

cout << " O ";

TextColor(10);

cout << char(179);

} else if (maze[m-1][n-1] == -1) {

TextColor(4);

cout << " X ";

TextColor(10);

cout << char(179);

}

else {

cout << " ";

TextColor(13);

cout << maze[m - 1][n - 1];

cout << " ";

TextColor(10);

cout << char(179);

}

cout << endl;

TextColor(10);

cout << char(192);

for (int i = 0; i < n - 1; i++) {

cout << char(196) << char(196) << char(196) << char(193);

}

cout << char(196) << char(196) << char(196) << char(217);

//Score cua 2 Robots

gotoxy(87, 6);

TextColor(4);

cout << score1;

gotoxy(87, 15);

TextColor(1);

cout << score2;

}

void calculateWinner(robot RobotA, robot RobotB){

gotoxy(75, 25);

for(int i=0; i<15;i++){

TextColor(i);

gotoxy(83, 23);

cout << "GAME OVER!";

Sleep(100);

hideCursor();

if (RobotA.score > RobotB.score) {

gotoxy(75, 25);

TextColor(4);

cout << "Robot A thang voi so diem " << RobotA.score << "!";

} else if (RobotB.score > RobotA.score) {

gotoxy(75, 25);

TextColor(9);

cout << "Robot B thang voi so diem " << RobotB.score << "!";

} else {

gotoxy(75, 25);

TextColor(7);

cout << "Hoa nhau! Ca hai Robot co cung so diem" << RobotA.score << "!";

}

}

TextColor(7);

gotoxy(83, 23);

cout << "GAME OVER!";

}

void move(robot& RobotA, robot& RobotB, vector<vector<int> >& maze, int numRows, int numCols) {

int currentX, currentY, newX, newY;

// Robot A's turn

currentX = RobotA.x;

currentY = RobotA.y;

// Find the maximum value among the neighboring cells

int maxScore = -1;

int maxIndex = -1;

for (int i = 0; i < 4; i++) {

newX = currentX + direction\_x[i];

newY = currentY + direction\_y[i];

if (isMoveValid(currentX, currentY, newX, newY, maze, RobotA.x, RobotA.y, RobotB.x, RobotB.y)) {

if (maze[newX][newY] > maxScore) {

maxScore = maze[newX][newY];

maxIndex = i;

}

}

}

// Move Robot A to the cell with the maximum score

if (maxIndex != -1) {

newX = currentX + direction\_x[maxIndex];

newY = currentY + direction\_y[maxIndex];

RobotA.x = newX;

RobotA.y = newY;

RobotA.steps++;

gotoxy(93, 8);

TextColor(4);

cout << RobotA.steps;

RobotA.score += maze[newX][newY];

RobotA.path.push\_back(maze[newX][newY]);

maze[currentX][currentY] = -1;

maze[newX][newY] = -1;

}

if (maxIndex == -1 || maxIndex == -2){

calculateWinner(RobotA, RobotB);

return;

}

// Print the maze after Robot A's move

veBanCo(maze, RobotA.x, RobotA.y, RobotB.x, RobotB.y, RobotA.score, RobotB.score);

// Robot B's turn

currentX = RobotB.x;

currentY = RobotB.y;

// Find the maximum value among the neighboring cells

maxScore = -2;

maxIndex = -2;

for (int i = 0; i < 4; i++) {

newX = currentX + direction\_x[i];

newY = currentY + direction\_y[i];

if (isMoveValid(currentX, currentY, newX, newY, maze, RobotA.x, RobotA.y, RobotB.x, RobotB.y)) {

if (maze[newX][newY] > maxScore) {

maxScore = maze[newX][newY];

maxIndex = i;

}

}

}

// Move Robot B to the cell with the maximum score

if (maxIndex != -2) {

newX = currentX + direction\_x[maxIndex];

newY = currentY + direction\_y[maxIndex];

RobotB.x = newX;

RobotB.y = newY;

RobotB.steps++;

gotoxy(93, 17);

TextColor(1);

cout << RobotB.steps;

RobotB.score += maze[newX][newY];

RobotB.path.push\_back(maze[newX][newY]);

maze[currentX][currentY] = -2;

maze[newX][newY] = -2;

}

// Print the maze after Robot B's move

veBanCo(maze, RobotA.x, RobotA.y, RobotB.x, RobotB.y, RobotA.score, RobotB.score);

// Recursive call for the next round

move(RobotA, RobotB, maze, numRows, numCols);

}

int main(){

vector<vector<int> > maze(numRows, vector<int>(numCols));

srand(time(0));

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

for (int j = 0; j < numCols; j++) {

maze[i][j] = rand() % 9 + 1;

}

}

//Khoi tao Me Cung luc dau

khoitaoBanCo(maze);

// Ve vien

TextColor(14);

gotoxy(57, 0); cout << char(201);

for(int i=0; i<30; i++){

gotoxy(57, i);

cout << char(186);

Sleep(1);

}

gotoxy(57, 0); cout << char(201);

gotoxy(57, 29); cout << char(200);

for(int i=0; i < 62; i++){

gotoxy(i+58, 29); cout << char(205);

Sleep(1);

}

gotoxy(119, 29); cout << char(188);

for(int i = 58; i < 119; i++){

gotoxy(i, 0); cout << char(205);

Sleep(1);

}

gotoxy(119, 0); cout << char(187);

for(int i = 1 ; i < 29; i++){

gotoxy(119, i); cout << char(186);

Sleep(1);

}

//Khung Player1

gotoxy(118, 6); cout << char(219);

for(int i=117; i >= 103; i--){

gotoxy(i, 6) ;cout << char(196);

Sleep(1);

}

gotoxy(102, 6); cout << char(254);

for(int i=6; i >= 2; i--){

gotoxy(101, i) ;cout << char(219);

Sleep(1);

}

for(int i = 101; i >= 73; i--){

gotoxy(i, 2); cout << char(220);

Sleep(1);

}

for(int i=3; i <=10; i++){

gotoxy(73, i); cout << char(219);

Sleep(1);

}

for(int i=73; i<=101; i++){

gotoxy(i, 10); cout << char(223);

Sleep(1);

}

for(int i=9; i>=6; i--){

gotoxy(101, i); cout << char(219);

Sleep(1);

}

//Khung Player2

gotoxy(118, 15); cout << char(219);

for(int i=117; i >= 103; i--){

gotoxy(i, 15) ;cout << char(196);

Sleep(1);

}

gotoxy(102, 15); cout << char(254);

for(int i=15; i >= 11; i--){

gotoxy(101, i) ;cout << char(219);

Sleep(1);

}

for(int i = 101; i >= 73; i--){

gotoxy(i, 11); cout << char(220);

Sleep(1);

}

for(int i=12; i <=19; i++){

gotoxy(73, i); cout << char(219);

Sleep(1);

}

for(int i=73; i<=101; i++){

gotoxy(i, 19); cout << char(223);

Sleep(1);

}

for(int i=18; i>=15; i--){

gotoxy(101, i); cout << char(219);

Sleep(1);

}

//Viet vao khung cua nguoi choi 1

gotoxy(83, 4);

for(int i=0; i<15;i++){

TextColor(i);

gotoxy(83, 4);

cout << "PLAYER 1";

Sleep(20);

hideCursor();

}

gotoxy(83, 4);

TextColor(4);

cout << "PLAYER 1";

gotoxy(77, 6);

TextColor(7);

cout << "Score A : ";

Sleep(100);

gotoxy(77, 8);

TextColor(7);

cout << "Steps Robot A : ";

Sleep(100);

//Viet vao khung cua nguoi choi 2

gotoxy(83, 13);

for(int i=0; i<15;i++){

TextColor(i);

gotoxy(83, 13);

cout << "PLAYER 2";

Sleep(20);

hideCursor();

}

gotoxy(83, 13);

TextColor(1);

cout << "PLAYER 2";

gotoxy(77, 15);

TextColor(7);

cout << "Score B : ";

Sleep(100);

gotoxy(77, 17);

TextColor(7);

cout << "Steps Robot B : ";

Sleep(100);

//KetQua 2 Robot

TextColor(14);

gotoxy(119, 21); cout << char(185);

gotoxy(57, 21); cout << char(204);

for(int i=58; i<=118; i++){

gotoxy(i, 21); cout << char(205);

Sleep(1);

}

//Nhap vi tri 2 Robots

bool passed[numRows][numCols] = {false};

robot RobotA;

robot RobotB;

//Step ban dau 2 robots

RobotA.steps = 0;

gotoxy(93, 8);

TextColor(4);

cout << RobotA.steps;

RobotB.steps = 0;

gotoxy(93, 17);

TextColor(1);

cout << RobotB.steps;

//Vi tri bat dau cua Robots

gotoxy(75, 23);

for(int i=0; i<15;i++){

TextColor(i);

gotoxy(75, 23);

cout << "Vi tri bat dau cua Robots";

Sleep(40);

hideCursor();

}

gotoxy(75, 23);

TextColor(11);

cout << "Vi tri bat dau cua Robots";

TextColor(7);

gotoxy(72,25);

cout << "->";

gotoxy(75, 25);

TextColor(4);

cout << "Robot A: ";

cin >> RobotA.x >> RobotA.y;

gotoxy(72,25);

cout <<" ";

RobotA.path.push\_back(maze[RobotA.x][RobotA.y]);

TextColor(7);

gotoxy(72,27);

cout << "->";

gotoxy(75, 27);

TextColor(1);

cout << "Robot B: ";

cin >> RobotB.x >> RobotB.y;

gotoxy(72,27);

cout <<" ";

RobotB.path.push\_back(maze[RobotB.x][RobotB.y]);

RobotA.score = maze[RobotA.x][RobotA.y];

//Diem ban dau robot A

gotoxy(87, 6);

TextColor(4);

cout << RobotA.score;

RobotB.score = maze[RobotB.x][RobotB.y];

//Diem ban dau robot B

gotoxy(87, 15);

TextColor(1);

cout << RobotB.score;

maze[RobotA.x][RobotA.y] = -1;

maze[RobotB.x][RobotB.y] = -2;

//Xoa nhap vi tri

for(int i=75; i<101; i++){

gotoxy(i, 23);

cout << char(32);

Sleep(2);

}

for(int i=72; i<92; i++){

gotoxy(i, 25);

cout << char(32);

Sleep(2);

}

for(int i=72; i<92; i++){

gotoxy(i, 27);

cout << char(32);

Sleep(2);

}

veBanCo(maze, RobotA.x, RobotA.y, RobotB.x, RobotB.y, RobotA.score, RobotB.score);

move(RobotA, RobotB, maze, numRows, numCols);

return 0;

}

MENU INTERFACE

#include <iostream>

#include "console.h"

#include <windows.h>

#include <cstdlib>

#include <stdio.h>

#include <conio.h>

#include <ctime>

#include <vector>

#include <fstream>

#include <mmsystem.h>

using namespace std;

int numRows = 14;

int numCols = 14;

int direction\_x[4] = {-1, 1, 0, 0};

int direction\_y[4] = {0, 0, -1, 1};

using namespace std;

struct robot {

int x, y;

int steps;

int score;

vector<int> path;

};

void gotoxy (int column, int line){

COORD coord;

coord.X = column;

coord.Y = line;

SetConsoleCursorPosition(GetStdHandle(STD\_OUTPUT\_HANDLE),coord);

}

void TextColor (int color){

SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE) , color);

}

void clrscr(){

CONSOLE\_SCREEN\_BUFFER\_INFO csbiInfo;

HANDLE hConsoleOut;

COORD Home = {0,0};

DWORD dummy;

hConsoleOut = GetStdHandle(STD\_OUTPUT\_HANDLE);

GetConsoleScreenBufferInfo(hConsoleOut,&csbiInfo);

FillConsoleOutputCharacter(hConsoleOut,' ',csbiInfo.dwSize.X \* csbiInfo.dwSize.Y,Home,&dummy);

csbiInfo.dwCursorPosition.X = 0;

csbiInfo.dwCursorPosition.Y = 0;

SetConsoleCursorPosition(hConsoleOut,csbiInfo.dwCursorPosition);

}

void hideCursor(){

HANDLE consoleHandle = GetStdHandle(STD\_OUTPUT\_HANDLE);

CONSOLE\_CURSOR\_INFO cursorInfo;

GetConsoleCursorInfo(consoleHandle, &cursorInfo);

cursorInfo.bVisible = false;

SetConsoleCursorInfo(consoleHandle, &cursorInfo);

}

void khoitaoBanCo(const vector<vector<int> >& maze) {

int m = maze.size();

int n = maze[0].size();

TextColor(10);

cout << char(218);

for (int i = 0; i < n - 1; i++) {

cout << char(196) << char(196) << char(196) << char(194);

}

cout << char(196) << char(196) << char(196) << char(191) << endl;

for (int i = 0; i < m - 1; i++) {

cout << char(179);

for (int k = 0; k < n - 1; k++) {

cout << " ";

TextColor(13);

cout << maze[i][k];

cout << " ";

TextColor(10);

cout << char(179);

}

cout << " ";

TextColor(13);

cout << maze[i][n - 1];

cout << " ";

TextColor(10);

cout << char(179) << endl;

cout << char(195);

for (int j = 0; j < n - 1; j++) {

cout << char(196) << char(196) << char(196) << char(197);

}

cout << char(196) << char(196) << char(196) << char(180) << endl;

}

cout << char(179);

for (int k = 0; k < n - 1; k++) {

cout << " ";

TextColor(13);

cout << maze[m - 1][k];

cout << " ";

TextColor(10);

cout << char(179);

}

cout << " ";

TextColor(13);

cout << maze[m - 1][n - 1];

cout << " ";

TextColor(10);

cout << char(179) << endl;

cout << char(192);

for (int i = 0; i < n - 1; i++) {

cout << char(196) << char(196) << char(196) << char(193);

}

cout << char(196) << char(196) << char(196) << char(217);

}

void taoBanCoIntersection1(const vector<vector<int> >& intersect1) {

int m = intersect1.size();

int n = intersect1[0].size();

TextColor(11);

cout << char(218);

for (int i = 0; i < n - 1; i++) {

cout << char(196) << char(196) << char(196) << char(194);

}

cout << char(196) << char(196) << char(196) << char(191) << endl;

for (int i = 0; i < m - 1; i++) {

cout << char(179);

for (int k = 0; k < n - 1; k++) {

cout << " ";

TextColor(10);

cout << intersect1[i][k];

cout << " ";

TextColor(11);

cout << char(179);

}

cout << " ";

TextColor(10);

cout << intersect1[i][n - 1];

cout << " ";

TextColor(11);

cout << char(179) << endl;

cout << char(195);

for (int j = 0; j < n - 1; j++) {

cout << char(196) << char(196) << char(196) << char(197);

}

cout << char(196) << char(196) << char(196) << char(180) << endl;

}

cout << char(179);

for (int k = 0; k < n - 1; k++) {

cout << " ";

TextColor(10);

cout << intersect1[m - 1][k];

cout << " ";

TextColor(11);

cout << char(179);

}

cout << " ";

TextColor(10);

cout << intersect1[m - 1][n - 1];

cout << " ";

TextColor(11);

cout << char(179) << endl;

cout << char(192);

for (int i = 0; i < n - 1; i++) {

cout << char(196) << char(196) << char(196) << char(193);

}

cout << char(196) << char(196) << char(196) << char(217);

}

void taoBanCoIntersection2(const vector<vector<int> >& intersect2) {

int m = intersect2.size();

int n = intersect2[0].size();

gotoxy(0, 15);

TextColor(12);

cout << char(218);

for (int i = 0; i < n - 1; i++) {

cout << char(196) << char(196) << char(196) << char(194);

}

cout << char(196) << char(196) << char(196) << char(191) << endl;

for (int i = 0; i < m - 1; i++) {

cout << char(179);

for (int k = 0; k < n - 1; k++) {

cout << " ";

TextColor(10);

cout << intersect2[i][k];

cout << " ";

TextColor(12);

cout << char(179);

}

cout << " ";

TextColor(10);

cout << intersect2[i][n - 1];

cout << " ";

TextColor(12);

cout << char(179) << endl;

cout << char(195);

for (int j = 0; j < n - 1; j++) {

cout << char(196) << char(196) << char(196) << char(197);

}

cout << char(196) << char(196) << char(196) << char(180) << endl;

}

cout << char(179);

for (int k = 0; k < n - 1; k++) {

cout << " ";

TextColor(10);

cout << intersect2[m - 1][k];

cout << " ";

TextColor(12);

cout << char(179);

}

cout << " ";

TextColor(10);

cout << intersect2[m - 1][n - 1];

cout << " ";

TextColor(12);

cout << char(179) << endl;

cout << char(192);

for (int i = 0; i < n - 1; i++) {

cout << char(196) << char(196) << char(196) << char(193);

}

cout << char(196) << char(196) << char(196) << char(217);

}

bool isMoveValid(int currentX, int currentY, int newX, int newY, vector<vector<int> > maze, int robot1X, int robot1Y, int robot2X, int robot2Y) {

if (newX < 0 || newX >= maze.size() || newY < 0 || newY >= maze[0].size()) {

// Robot has gone outside the maze!

return false;

}

if (maze[newX][newY] == -1 || maze[newX][newY] == -2) {

// This position has been visited before!

return false;

}

if ((newX == robot1X && newY == robot1Y) || (newX == robot2X && newY == robot2Y)) {

// Robot cannot move into the other robot's cell!

return false;

}

return true;

}

bool isMoveValid\_Intersect1(int currentX, int currentY, int newX, int newY, vector<vector<int> > intersect1) {

if (newX < 0 || newX >= intersect1.size() || newY < 0 || newY >= intersect1[0].size()) {

// Robot has gone outside the maze!

return false;

}

if (intersect1[newX][newY] == -1) {

return false;

}

return true;

}

bool isMoveValid\_Intersect2(int currentX, int currentY, int newX, int newY, vector<vector<int> > intersect2) {

if (newX < 0 || newX >= intersect2.size() || newY < 0 || newY >= intersect2[0].size()) {

// Robot has gone outside the maze!

return false;

}

if (intersect2[newX][newY] == -2) {

return false;

}

return true;

}

void veBanCo(const vector<vector<int> >& maze, int robot1X, int robot1Y, int robot2X, int robot2Y, int score1, int score2) {

gotoxy(0,0);

TextColor(10);

int m = maze.size();

int n = maze[0].size();

cout << char(218);

for (int i = 0; i < n - 1; i++) {

cout << char(196) << char(196) << char(196) << char(194);

}

cout << char(196) << char(196) << char(196) << char(191) << endl;

for (int i = 0; i < m - 1; i++) {

cout << char(179);

for (int k = 0; k < n - 1; k++) {

if (i == robot1X && k == robot1Y) {

TextColor(7);

cout << " A ";

TextColor(10);

cout << char(179);

} else if (i == robot2X && k == robot2Y) {

TextColor(7);

cout << " B ";

TextColor(10);

cout << char(179);

} else if (maze[i][k] == -2) {

TextColor(1);

cout << " O ";

TextColor(10);

cout << char(179);

} else if (maze[i][k] == -1) {

TextColor(4);

cout << " X ";

TextColor(10);

cout << char(179);

} else {

cout << " ";

TextColor(13);

cout << maze[i][k];

cout << " ";

TextColor(10);

cout << char(179);

}

}

if (i == robot1X && n - 1 == robot1Y) {

TextColor(7);

cout << " A ";

TextColor(10);

cout << char(179);

} else if (i == robot2X && n - 1 == robot2Y) {

TextColor(7);

cout << " B ";

TextColor(10);

cout << char(179);

} else if (maze[i][n-1] == -2) {

TextColor(1);

cout << " O ";

TextColor(10);

cout << char(179);

} else if (maze[i][n-1] == -1) {

TextColor(4);

cout << " X ";

TextColor(10);

cout << char(179);

} else {

cout << " ";

TextColor(13);

cout << maze[i][n - 1];

cout << " ";

TextColor(10);

cout << char(179);

}

cout << endl;

TextColor(10);

cout << char(195);

for (int j = 0; j < n - 1; j++) {

cout << char(196) << char(196) << char(196) << char(197);

}

cout << char(196) << char(196) << char(196) << char(180) << endl;

}

cout << char(179);

for (int k = 0; k < n - 1; k++) {

if (m - 1 == robot1X && k == robot1Y) {

TextColor(7);

cout << " A ";

TextColor(10);

cout << char(179);

} else if (m - 1 == robot2X && k == robot2Y) {

TextColor(7);

cout << " B ";

TextColor(10);

cout << char(179);

} else if (maze[m-1][k] == -2) {

TextColor(1);

cout << " O ";

TextColor(10);

cout << char(179);

} else if (maze[m-1][k] == -1) {

TextColor(4);

cout << " X ";

TextColor(10);

cout << char(179);

}

else {

cout << " ";

TextColor(13);

cout << maze[m - 1][k];

cout << " ";

TextColor(10);

cout << char(179);

}

}

if (m - 1 == robot1X && n - 1 == robot1Y) {

TextColor(7);

cout << " A ";

TextColor(10);

cout << char(179);

} else if (m - 1 == robot2X && n - 1 == robot2Y) {

TextColor(7);

cout << " B ";

TextColor(10);

cout << char(179);

} else if (maze[m-1][n-1] == -2) {

TextColor(1);

cout << " O ";

TextColor(10);

cout << char(179);

} else if (maze[m-1][n-1] == -1) {

TextColor(4);

cout << " X ";

TextColor(10);

cout << char(179);

}

else {

cout << " ";

TextColor(13);

cout << maze[m - 1][n - 1];

cout << " ";

TextColor(10);

cout << char(179);

}

cout << endl;

TextColor(10);

cout << char(192);

for (int i = 0; i < n - 1; i++) {

cout << char(196) << char(196) << char(196) << char(193);

}

cout << char(196) << char(196) << char(196) << char(217);

//Score cua 2 Robots

gotoxy(87, 6);

TextColor(4);

cout << score1;

gotoxy(87, 15);

TextColor(1);

cout << score2;

}

void veBanCoInterSection1(const vector<vector<int> >& intersect1, int robot1X, int robot1Y, int score1) {

gotoxy(0,0);

TextColor(11);

int m = intersect1.size();

int n = intersect1[0].size();

cout << char(218);

for (int i = 0; i < n - 1; i++) {

cout << char(196) << char(196) << char(196) << char(194);

}

cout << char(196) << char(196) << char(196) << char(191) << endl;

for (int i = 0; i < m - 1; i++) {

cout << char(179);

for (int k = 0; k < n - 1; k++) {

if (i == robot1X && k == robot1Y) {

TextColor(7);

cout << " A ";

TextColor(11);

cout << char(179);

} else if (intersect1[i][k] == -1) {

TextColor(4);

cout << " X ";

TextColor(11);

cout << char(179);

} else {

cout << " ";

TextColor(10);

cout << intersect1[i][k];

cout << " ";

TextColor(11);

cout << char(179);

}

}

if (i == robot1X && n - 1 == robot1Y) {

TextColor(7);

cout << " A ";

TextColor(11);

cout << char(179);

} else if (intersect1[i][n-1] == -1) {

TextColor(4);

cout << " X ";

TextColor(11);

cout << char(179);

} else {

cout << " ";

TextColor(10);

cout << intersect1[i][n - 1];

cout << " ";

TextColor(11);

cout << char(179);

}

cout << endl;

TextColor(11);

cout << char(195);

for (int j = 0; j < n - 1; j++) {

cout << char(196) << char(196) << char(196) << char(197);

}

cout << char(196) << char(196) << char(196) << char(180) << endl;

}

cout << char(179);

for (int k = 0; k < n - 1; k++) {

if (m - 1 == robot1X && k == robot1Y) {

TextColor(7);

cout << " A ";

TextColor(11);

cout << char(179);

} else if (intersect1[m-1][k] == -1) {

TextColor(4);

cout << " X ";

TextColor(11);

cout << char(179);

}

else {

cout << " ";

TextColor(10);

cout << intersect1[m - 1][k];

cout << " ";

TextColor(11);

cout << char(179);

}

}

if (m - 1 == robot1X && n - 1 == robot1Y) {

TextColor(7);

cout << " A ";

TextColor(11);

cout << char(179);

} else if (intersect1[m-1][n-1] == -1) {

TextColor(4);

cout << " X ";

TextColor(11);

cout << char(179);

}

else {

cout << " ";

TextColor(10);

cout << intersect1[m - 1][n - 1];

cout << " ";

TextColor(11);

cout << char(179);

}

cout << endl;

TextColor(11);

cout << char(192);

for (int i = 0; i < n - 1; i++) {

cout << char(196) << char(196) << char(196) << char(193);

}

cout << char(196) << char(196) << char(196) << char(217);

//Score cua 2 Robots

gotoxy(87, 6);

TextColor(4);

cout << score1;

}

void veBanCoInterSection2(const vector<vector<int> >& intersect2, int robot2X, int robot2Y, int score2) {

gotoxy(0,15);

TextColor(12);

int m = intersect2.size();

int n = intersect2[0].size();

cout << char(218);

for (int i = 0; i < n - 1; i++) {

cout << char(196) << char(196) << char(196) << char(194);

}

cout << char(196) << char(196) << char(196) << char(191) << endl;

for (int i = 0; i < m - 1; i++) {

cout << char(179);

for (int k = 0; k < n - 1; k++) {

if (i == robot2X && k == robot2Y) {

TextColor(7);

cout << " B ";

TextColor(12);

cout << char(179);

} else if (intersect2[i][k] == -2) {

TextColor(1);

cout << " O ";

TextColor(12);

cout << char(179);

} else {

cout << " ";

TextColor(10);

cout << intersect2[i][k];

cout << " ";

TextColor(12);

cout << char(179);

}

}

if (i == robot2X && n - 1 == robot2Y) {

TextColor(7);

cout << " B ";

TextColor(12);

cout << char(179);

} else if (intersect2[i][n-1] == -2) {

TextColor(1);

cout << " O ";

TextColor(12);

cout << char(179);

} else {

cout << " ";

TextColor(10);

cout << intersect2[i][n - 1];

cout << " ";

TextColor(12);

cout << char(179);

}

cout << endl;

TextColor(12);

cout << char(195);

for (int j = 0; j < n - 1; j++) {

cout << char(196) << char(196) << char(196) << char(197);

}

cout << char(196) << char(196) << char(196) << char(180) << endl;

}

cout << char(179);

for (int k = 0; k < n - 1; k++) {

if (m - 1 == robot2X && k == robot2Y) {

TextColor(7);

cout << " B ";

TextColor(12);

cout << char(179);

} else if (intersect2[m-1][k] == -2) {

TextColor(1);

cout << " O ";

TextColor(12);

cout << char(179);

}

else {

cout << " ";

TextColor(10);

cout << intersect2[m - 1][k];

cout << " ";

TextColor(12);

cout << char(179);

}

}

if (m - 1 == robot2X && n - 1 == robot2Y) {

TextColor(7);

cout << " B ";

TextColor(12);

cout << char(179);

} else if (intersect2[m-1][n-1] == -2) {

TextColor(1);

cout << " O ";

TextColor(12);

cout << char(179);

}

else {

cout << " ";

TextColor(10);

cout << intersect2[m - 1][n - 1];

cout << " ";

TextColor(12);

cout << char(179);

}

cout << endl;

TextColor(12);

cout << char(192);

for (int i = 0; i < n - 1; i++) {

cout << char(196) << char(196) << char(196) << char(193);

}

cout << char(196) << char(196) << char(196) << char(217);

//Score cua Robots 2

gotoxy(87, 15);

TextColor(1);

cout << score2;

}

void moveIntersection2(robot& RobotB, vector<vector<int> >& intersect2, int numRows, int numCols) {

int currentX, currentY, newX, newY;

// Robot B's turn

currentX = RobotB.x;

currentY = RobotB.y;

// Find the maximum value among the neighboring cells

int maxScore = -2;

int maxIndex = -2;

for (int i = 0; i < 4; i++) {

newX = currentX + direction\_x[i];

newY = currentY + direction\_y[i];

if (isMoveValid\_Intersect2(currentX, currentY, newX, newY, intersect2)) {

if (intersect2[newX][newY] > maxScore) {

maxScore = intersect2[newX][newY];

maxIndex = i;

}

}

}

// Move Robot A to the cell with the maximum score

if (maxIndex != -2) {

newX = currentX + direction\_x[maxIndex];

newY = currentY + direction\_y[maxIndex];

RobotB.x = newX;

RobotB.y = newY;

RobotB.steps++;

gotoxy(93, 17);

TextColor(1);

cout << RobotB.steps;

RobotB.score += intersect2[newX][newY];

RobotB.path.push\_back(intersect2[newX][newY]);

intersect2[currentX][currentY] = -2;

intersect2[newX][newY] = -2;

}

if (maxIndex == -2){

return;

}

veBanCoInterSection2(intersect2, RobotB.x, RobotB.y, RobotB.score);

Sleep(100);

moveIntersection2(RobotB, intersect2, numRows, numCols);

}

void moveIntersection1(robot& RobotA, vector<vector<int> >& intersect1, int numRows, int numCols) {

int currentX, currentY, newX, newY;

// Robot A's turn

currentX = RobotA.x;

currentY = RobotA.y;

// Find the maximum value among the neighboring cells

int maxScore = -1;

int maxIndex = -1;

for (int i = 0; i < 4; i++) {

newX = currentX + direction\_x[i];

newY = currentY + direction\_y[i];

if (isMoveValid\_Intersect1(currentX, currentY, newX, newY, intersect1)) {

if (intersect1[newX][newY] > maxScore) {

maxScore = intersect1[newX][newY];

maxIndex = i;

}

}

}

// Move Robot A to the cell with the maximum score

if (maxIndex != -1) {

newX = currentX + direction\_x[maxIndex];

newY = currentY + direction\_y[maxIndex];

RobotA.x = newX;

RobotA.y = newY;

RobotA.steps++;

gotoxy(93, 8);

TextColor(4);

cout << RobotA.steps;

RobotA.score += intersect1[newX][newY];

RobotA.path.push\_back(intersect1[newX][newY]);

intersect1[currentX][currentY] = -1;

intersect1[newX][newY] = -1;

}

if (maxIndex == -1){

return;

}

veBanCoInterSection1(intersect1, RobotA.x, RobotA.y, RobotA.score);

Sleep(100);

moveIntersection1(RobotA, intersect1, numRows, numCols);

}

void P1WIN(){

PlaySound(TEXT("win.wav"), NULL, SND\_FILENAME | SND\_ASYNC);

gotoxy(64, 22);

for(int i=1; i < 15; i++){

TextColor(i);

gotoxy(64, 22);

cout << " \_\_\_\_\_\_\_ \_\_ \_\_\_\_ \_\_\_\_ \_\_\_\_\_ \_\_\_\_ \_\_\_\_\_ " << endl; gotoxy(64, 23);

cout << "|\_ \_\_ \\/ | |\_ \_| |\_ \_|\_ \_|\_ \\|\_ \_| " << endl; gotoxy(64, 24);

cout << " | |\_\_) `| | \\ \\ /\\ / / | | | \\ | | " << endl; gotoxy(64, 25);

cout << " | \_\_\_/ | | \\ \\/ \\/ / | | | |\\ \\| | " << endl; gotoxy(64, 26);

cout << " \_| |\_ \_| |\_ \\ /\\ / \_| |\_ \_| |\_\\ |\_ " << endl; gotoxy(64, 27);

cout << "|\_\_\_\_\_| |\_\_\_\_\_| \\/ \\/ |\_\_\_\_\_|\_\_\_\_\_|\\\_\_\_\_| " <<endl;

Sleep(100);

}

TextColor(4);

gotoxy(64, 22);

cout << " \_\_\_\_\_\_\_ \_\_ \_\_\_\_ \_\_\_\_ \_\_\_\_\_ \_\_\_\_ \_\_\_\_\_ " << endl; gotoxy(64, 23);

cout << "|\_ \_\_ \\/ | |\_ \_| |\_ \_|\_ \_|\_ \\|\_ \_| " << endl; gotoxy(64, 24);

cout << " | |\_\_) `| | \\ \\ /\\ / / | | | \\ | | " << endl; gotoxy(64, 25);

cout << " | \_\_\_/ | | \\ \\/ \\/ / | | | |\\ \\| | " << endl; gotoxy(64, 26);

cout << " \_| |\_ \_| |\_ \\ /\\ / \_| |\_ \_| |\_\\ |\_ " << endl; gotoxy(64, 27);

cout << "|\_\_\_\_\_| |\_\_\_\_\_| \\/ \\/ |\_\_\_\_\_|\_\_\_\_\_|\\\_\_\_\_| " <<endl;

}

void P2WIN(){

gotoxy(64, 22);

PlaySound(TEXT("win.wav"), NULL, SND\_FILENAME | SND\_ASYNC);

for(int i=1; i < 15; i++){

TextColor(i);

gotoxy(64, 22);

cout << " \_\_\_\_\_\_\_ \_\_\_\_\_ \_\_\_\_ \_\_\_\_ \_\_\_\_\_ \_\_\_\_ \_\_\_\_\_ " << endl; gotoxy(64, 23);

cout << "|\_ \_\_ \\ / \_\_\_ `. |\_ \_| |\_ \_|\_ \_|\_ \\|\_ \_| " << endl; gotoxy(64, 24);

cout << " | |\_\_) |\_/\_\_\_) | \\ \\ /\\ / / | | | \\ | | " << endl; gotoxy(64, 25);

cout << " | \_\_\_/ .'\_\_\_\_.' \\ \\/ \\/ / | | | |\\ \\| | " << endl; gotoxy(64, 26);

cout << " \_| |\_ / /\_\_\_\_\_ \\ /\\ / \_| |\_ \_| | \\ |\_ " << endl; gotoxy(64, 27);

cout << "|\_\_\_\_\_| |\_\_\_\_\_\_\_| \\/ \\/ |\_\_\_\_\_|\_\_\_\_\_|\\_\_\_\_| " << endl;

Sleep(100);

}

TextColor(1);

gotoxy(64, 22);

cout << " \_\_\_\_\_\_\_ \_\_\_\_\_ \_\_\_\_ \_\_\_\_ \_\_\_\_\_ \_\_\_\_ \_\_\_\_\_ " << endl; gotoxy(64, 23);

cout << "|\_ \_\_ \\ / \_\_\_ `. |\_ \_| |\_ \_|\_ \_|\_ \\|\_ \_| " << endl; gotoxy(64, 24);

cout << " | |\_\_) |\_/\_\_\_) | \\ \\ /\\ / / | | | \\ | | " << endl; gotoxy(64, 25);

cout << " | \_\_\_/ .'\_\_\_\_.' \\ \\/ \\/ / | | | |\\ \\| | " << endl; gotoxy(64, 26);

cout << " \_| |\_ / /\_\_\_\_\_ \\ /\\ / \_| |\_ \_| | \\ |\_ " << endl; gotoxy(64, 27);

cout << "|\_\_\_\_\_| |\_\_\_\_\_\_\_| \\/ \\/ |\_\_\_\_\_|\_\_\_\_\_|\\_\_\_\_| " << endl;

}

void DRAW(){

gotoxy(66, 22);

PlaySound(TEXT("win.wav"), NULL, SND\_FILENAME | SND\_ASYNC);

for(int i=1; i < 15; i++){

TextColor(i);

gotoxy(66, 22);

cout << " \_\_\_\_\_\_ \_\_\_\_\_\_\_ \_ \_\_\_\_ \_\_\_\_ " << endl; gotoxy(66, 23);

cout << "|\_ \_ `|\_ \_\_ \\ / |\_ \_| |\_ \_|" << endl; gotoxy(66, 24);

cout << " | | `. \\| |\_\_) | / \_ \\ \\ /\\ / / " << endl; gotoxy(66, 25);

cout << " | | | || \_\_ / / \_\_\_ \\/\\ \\/ / " << endl; gotoxy(66, 26);

cout << " \_| |\_.' \_| | \\ \\\_ \_/ / \\ \\ /\\ / " << endl; gotoxy(66, 27);

cout << "|\_\_\_\_\_\_.|\_\_\_\_| |\_\_\_|\_\_\_\_| |\_\_\_\_\\/ \\/ " << endl;

Sleep(100);

}

TextColor(13);

gotoxy(66, 22);

cout << " \_\_\_\_\_\_ \_\_\_\_\_\_\_ \_ \_\_\_\_ \_\_\_\_ " << endl; gotoxy(66, 23);

cout << "|\_ \_ `|\_ \_\_ \\ / |\_ \_| |\_ \_|" << endl; gotoxy(66, 24);

cout << " | | `. \\| |\_\_) | / \_ \\ \\ /\\ / / " << endl; gotoxy(66, 25);

cout << " | | | || \_\_ / / \_\_\_ \\/\\ \\/ / " << endl; gotoxy(66, 26);

cout << " \_| |\_.' \_| | \\ \\\_ \_/ / \\ \\ /\\ / " << endl; gotoxy(66, 27);

cout << "|\_\_\_\_\_\_.|\_\_\_\_| |\_\_\_|\_\_\_\_| |\_\_\_\_\\/ \\/ " << endl;

}

void calculateWinner(robot RobotA, robot RobotB){

hideCursor();

if (RobotA.score > RobotB.score) {

P1WIN();

} else if (RobotB.score > RobotA.score) {

P2WIN();

} else {

DRAW();

}

//Luu duong di va so buoc cua 2 robots

ofstream outputFile("output.txt");

if (outputFile.is\_open()) {

outputFile << RobotA.steps <<endl;

for (int i = 0; i < RobotA.path.size(); i++) {

outputFile << RobotA.path[i] << " ";

}

outputFile << "\n\n" << RobotB.steps <<endl;

for (int i = 0; i < RobotB.path.size(); i++) {

outputFile << RobotB.path[i] << " ";

}

outputFile.close();

}

}

void move(robot& RobotA, robot& RobotB, vector<vector<int> >& maze, int numRows, int numCols) {

int currentX, currentY, newX, newY;

// Robot A's turn

currentX = RobotA.x;

currentY = RobotA.y;

// Find the maximum value among the neighboring cells

int maxScore = -1;

int maxIndex = -1;

for (int i = 0; i < 4; i++) {

newX = currentX + direction\_x[i];

newY = currentY + direction\_y[i];

if (isMoveValid(currentX, currentY, newX, newY, maze, RobotA.x, RobotA.y, RobotB.x, RobotB.y)) {

if (maze[newX][newY] > maxScore) {

maxScore = maze[newX][newY];

maxIndex = i;

}

}

}

// Move Robot A to the cell with the maximum score

if (maxIndex != -1) {

newX = currentX + direction\_x[maxIndex];

newY = currentY + direction\_y[maxIndex];

RobotA.x = newX;

RobotA.y = newY;

RobotA.steps++;

gotoxy(93, 8);

TextColor(4);

cout << RobotA.steps;

RobotA.score += maze[newX][newY];

RobotA.path.push\_back(maze[newX][newY]);

maze[currentX][currentY] = -1;

maze[newX][newY] = -1;

}

if (maxIndex == -1 || maxIndex == -2){

calculateWinner(RobotA, RobotB);

return;

}

// Print the maze after Robot A's move

veBanCo(maze, RobotA.x, RobotA.y, RobotB.x, RobotB.y, RobotA.score, RobotB.score);

Sleep(20);

// Robot B's turn

currentX = RobotB.x;

currentY = RobotB.y;

// Find the maximum value among the neighboring cells

maxScore = -2;

maxIndex = -2;

for (int i = 0; i < 4; i++) {

newX = currentX + direction\_x[i];

newY = currentY + direction\_y[i];

if (isMoveValid(currentX, currentY, newX, newY, maze, RobotA.x, RobotA.y, RobotB.x, RobotB.y)) {

if (maze[newX][newY] > maxScore) {

maxScore = maze[newX][newY];

maxIndex = i;

}

}

}

// Move Robot B to the cell with the maximum score

if (maxIndex != -2) {

newX = currentX + direction\_x[maxIndex];

newY = currentY + direction\_y[maxIndex];

RobotB.x = newX;

RobotB.y = newY;

RobotB.steps++;

gotoxy(93, 17);

TextColor(1);

cout << RobotB.steps;

RobotB.score += maze[newX][newY];

RobotB.path.push\_back(maze[newX][newY]);

maze[currentX][currentY] = -2;

maze[newX][newY] = -2;

}

// Print the maze after Robot B's move

veBanCo(maze, RobotA.x, RobotA.y, RobotB.x, RobotB.y, RobotA.score, RobotB.score);

Sleep(20);

// Recursive call for the next round

move(RobotA, RobotB, maze, numRows, numCols);

}

void exitGame(){

char ask;

int choice[] = {4,7};

int counter = 1;

char key;

gotoxy(44, 12);

TextColor(3);

cout << "Ban co chac chan muon thoat game?";

for(int i=0;;){

gotoxy(44,14);

TextColor(choice[0]);

cout<<"YES";

gotoxy(44,16);

TextColor(choice[1]);

cout<<"NO";

key = \_getch();

if(key == 72 && (counter >=2 && counter <= 5))

{

PlaySound(TEXT("go.wav"), NULL, SND\_FILENAME | SND\_ASYNC);

counter--;

}

if(key == 80 && (counter >=1 && counter <= 4))

{

PlaySound(TEXT("go.wav"), NULL, SND\_FILENAME | SND\_ASYNC);

counter++;

}

if(key == '\r')

{

if(counter == 1)

{

PlaySound(TEXT("tick.wav"), NULL, SND\_FILENAME | SND\_ASYNC);

clrscr();

for(int i=0; i<15; i++){

gotoxy(46, i);

TextColor(i);

cout << "Cam on da choi game!";

if(i==14){

break;

}

Sleep(70);

clrscr();

}

exit(0);

}

if(counter == 2)

{

PlaySound(TEXT("tick.wav"), NULL, SND\_FILENAME | SND\_ASYNC);

clrscr();

return;

}

}

choice[0] = 7;

choice[1] = 7;

if(counter == 1)

{

choice[0] = 4;

}

if(counter == 2)

{

choice[1] = 4;

}

}

}

void exitESC(){

while (true) {

if (\_kbhit()) {

char ch = \_getch();

if (ch == 27) {

return;

}

}

}

}

void khunggiaodien(){

TextColor(1);

gotoxy(0, 0);

cout << char(201);

for(int i=1; i<120; i++){

gotoxy(i, 0);

cout << char(205);

Sleep(1);

}

gotoxy(119, 0);

cout << char(187);

for(int i=1; i<30; i++){

gotoxy(119, i);

cout << char(186);

Sleep(1);

}

gotoxy(119, 29);

cout << char(188);

gotoxy(118, 29);

for(int i=118; i>=1; i--){

gotoxy(i, 29);

cout << char(205);

Sleep(1);

}

gotoxy(0, 29); cout << char(200);

for(int i=28; i>=1; i--){

gotoxy(0, i);

cout << char(186);

Sleep(1);

}

//Hai duong trang

for(int i = 46; i<73; i++){

TextColor(7);

gotoxy(i, 13);

cout << char(196);

Sleep(1);

}

for(int i = 46; i<73; i++){

TextColor(7);

gotoxy(i, 21);

cout << char(196);

Sleep(1);

}

}

void tenGame() {

PlaySound(TEXT("appear.wav"), NULL, SND\_FILENAME | SND\_ASYNC);

for(int i=1; i<15; i++){

TextColor(i);

gotoxy(21, 5);

cout << " \_\_\_ \_\_\_\_ \_\_\_\_ \_\_\_\_ \_\_\_\_ \_ \_ \_\_\_\_ \_\_ \_\_\_\_ \_\_ \_\_\_\_ \_\_\_\_ " << endl; gotoxy(21, 6);

cout << " / \_\_)( \_ \\( \_\_)( \_\_)( \\( \\/ ) ( \_ \\ / \\ ( \_ \\ / \\ (\_ \_)/ \_\_\_)" << endl; gotoxy(21, 7);

cout << "( (\_ \\ ) / ) \_) ) \_) ) D ( ) / ) /( O ) ) \_ (( O ) )( \\\_\_\_ \\" << endl; gotoxy(21, 8);

cout << " \\\_\_\_/(\_\_\\\_)(\_\_\_\_)(\_\_\_\_)(\_\_\_\_/(\_\_/ (\_\_\\\_) \\\_\_/ (\_\_\_\_/ \\\_\_/ (\_\_) (\_\_\_\_/" << endl;

Sleep(50);

}

TextColor(10);

gotoxy(21, 5);

cout << " \_\_\_ \_\_\_\_ \_\_\_\_ \_\_\_\_ \_\_\_\_ \_ \_ \_\_\_\_ \_\_ \_\_\_\_ \_\_ \_\_\_\_ \_\_\_\_ " << endl; gotoxy(21, 6);

cout << " / \_\_)( \_ \\( \_\_)( \_\_)( \\( \\/ ) ( \_ \\ / \\ ( \_ \\ / \\ (\_ \_)/ \_\_\_)" << endl; gotoxy(21, 7);

cout << "( (\_ \\ ) / ) \_) ) \_) ) D ( ) / ) /( O ) ) \_ (( O ) )( \\\_\_\_ \\" << endl; gotoxy(21, 8);

cout << " \\\_\_\_/(\_\_\\\_)(\_\_\_\_)(\_\_\_\_)(\_\_\_\_/(\_\_/ (\_\_\\\_) \\\_\_/ (\_\_\_\_/ \\\_\_/ (\_\_) (\_\_\_\_/" << endl;

}

void vientrongGame(){

TextColor(14);

gotoxy(57, 0); cout << char(201);

for(int i=0; i<30; i++){

gotoxy(57, i);

cout << char(186);

Sleep(1);

}

gotoxy(57, 0); cout << char(201);

gotoxy(57, 29); cout << char(200);

for(int i=0; i < 62; i++){

gotoxy(i+58, 29); cout << char(205);

Sleep(1);

}

gotoxy(119, 29); cout << char(188);

for(int i = 58; i < 119; i++){

gotoxy(i, 0); cout << char(205);

Sleep(1);

}

gotoxy(119, 0); cout << char(187);

for(int i = 1 ; i < 29; i++){

gotoxy(119, i); cout << char(186);

Sleep(1);

}

//Khung Player1

gotoxy(118, 6); cout << char(219);

for(int i=117; i >= 103; i--){

gotoxy(i, 6) ;cout << char(196);

Sleep(1);

}

gotoxy(102, 6); cout << char(254);

for(int i=6; i >= 2; i--){

gotoxy(101, i) ;cout << char(219);

Sleep(1);

}

for(int i = 101; i >= 73; i--){

gotoxy(i, 2); cout << char(220);

Sleep(1);

}

for(int i=3; i <=10; i++){

gotoxy(73, i); cout << char(219);

Sleep(1);

}

for(int i=73; i<=101; i++){

gotoxy(i, 10); cout << char(223);

Sleep(1);

}

for(int i=9; i>=6; i--){

gotoxy(101, i); cout << char(219);

Sleep(1);

}

//Khung Player2

gotoxy(118, 15); cout << char(219);

for(int i=117; i >= 103; i--){

gotoxy(i, 15) ;cout << char(196);

Sleep(1);

}

gotoxy(102, 15); cout << char(254);

for(int i=15; i >= 11; i--){

gotoxy(101, i) ;cout << char(219);

Sleep(1);

}

for(int i = 101; i >= 73; i--){

gotoxy(i, 11); cout << char(220);

Sleep(1);

}

for(int i=12; i <=19; i++){

gotoxy(73, i); cout << char(219);

Sleep(1);

}

for(int i=73; i<=101; i++){

gotoxy(i, 19); cout << char(223);

Sleep(1);

}

for(int i=18; i>=15; i--){

gotoxy(101, i); cout << char(219);

Sleep(1);

}

//Viet vao khung cua nguoi choi 1

gotoxy(83, 4);

for(int i=0; i<15;i++){

TextColor(i);

gotoxy(83, 4);

cout << "PLAYER 1";

Sleep(60);

hideCursor();

}

PlaySound(TEXT("tick.wav"), NULL, SND\_FILENAME | SND\_ASYNC);

gotoxy(83, 4);

TextColor(4);

cout << "PLAYER 1";

gotoxy(77, 6);

TextColor(7);

cout << "Score A : ";

Sleep(100);

gotoxy(77, 8);

TextColor(7);

cout << "Steps Robot A : ";

Sleep(100);

//Viet vao khung cua nguoi choi 2

gotoxy(83, 13);

for(int i=0; i<15;i++){

TextColor(i);

gotoxy(83, 13);

cout << "PLAYER 2";

Sleep(60);

hideCursor();

}

PlaySound(TEXT("tick.wav"), NULL, SND\_FILENAME | SND\_ASYNC);

gotoxy(83, 13);

TextColor(1);

cout << "PLAYER 2";

gotoxy(77, 15);

TextColor(7);

cout << "Score B : ";

Sleep(100);

gotoxy(77, 17);

TextColor(7);

cout << "Steps Robot B : ";

Sleep(100);

//KetQua 2 Robot

TextColor(14);

gotoxy(119, 21); cout << char(185);

gotoxy(57, 21); cout << char(204);

for(int i=58; i<=118; i++){

gotoxy(i, 21); cout << char(205);

Sleep(1);

}

}

void vitriCatNhau(robot& RobotA, robot& RobotB){

int count = 0;

for(int k=1; k<15; k++){

TextColor(k);

gotoxy(0, 0);

cout << "Hai robot cat nhau tai vi tri: ";

Sleep(20);

}

PlaySound(TEXT("tick.wav"), NULL, SND\_FILENAME | SND\_ASYNC);

TextColor(10);

gotoxy(0, 0);

cout << "Hai robot cat nhau tai vi tri: ";

for (size\_t i = 0; i < RobotA.path.size(); ++i) {

for (size\_t j = 0; j < RobotB.path.size(); ++j) {

if (RobotA.path[i] == RobotB.path[j]){

cout << RobotA.path[i] << " ";

count++;

break;

}

}

}

cout << "\nHai robot cat nhau tai " << count << " vi tri" << endl;

}

void ModeIntersection(){

hideCursor();

int numRows = 7; int numCols = 14;

robot RobotA;

robot RobotB;

srand(time(0));

vector<vector<int> > intersect1(numRows, vector<int>(numCols));

vector<vector<int> > intersect2(numRows, vector<int>(numCols));

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

for (int j = 0; j < numCols; j++) {

intersect1[i][j] = rand() % 9 + 1;

intersect2[i][j] = intersect1[i][j];

}

}

taoBanCoIntersection1(intersect1);

taoBanCoIntersection2(intersect2);

vientrongGame();

bool visited[numRows][numCols] = {false};

bool visited2[numRows][numCols] = {false};

//Step ban dau 2 robots

RobotA.steps = 0;

gotoxy(93, 8);

TextColor(4);

cout << RobotA.steps;

RobotB.steps = 0;

gotoxy(93, 17);

TextColor(1);

cout << RobotB.steps;

//Vi tri bat dau cua Robots

PlaySound(TEXT("appear.wav"), NULL, SND\_FILENAME | SND\_ASYNC);

gotoxy(75, 23);

for(int i=0; i<15;i++){

TextColor(i);

gotoxy(75, 23);

cout << "Vi tri bat dau cua Robots";

Sleep(40);

hideCursor();

}

gotoxy(75, 23);

TextColor(11);

cout << "Vi tri bat dau cua Robots";

TextColor(7);

gotoxy(72,25);

cout << "->";

gotoxy(75, 25);

TextColor(4);

cout << "Robot A: ";

cin >> RobotA.x >> RobotA.y;

RobotA.score = intersect1[RobotA.x][RobotA.y];

//Diem ban dau robot A

gotoxy(87, 6);

TextColor(4);

cout << RobotA.score;

PlaySound(TEXT("tick.wav"), NULL, SND\_FILENAME | SND\_ASYNC);

veBanCoInterSection1(intersect1, RobotA.x, RobotA.y, RobotA.score);

gotoxy(72,25);

cout <<" ";

TextColor(7);

gotoxy(72,27);

cout << "->";

gotoxy(75, 27);

TextColor(1);

cout << "Robot B: ";

cin >> RobotB.x >> RobotB.y;

RobotB.score = intersect2[RobotB.x][RobotB.y];

//Diem ban dau robot B

gotoxy(87, 15);

TextColor(1);

cout << RobotB.score;

PlaySound(TEXT("tick.wav"), NULL, SND\_FILENAME | SND\_ASYNC);

veBanCoInterSection2(intersect2, RobotB.x, RobotB.y, RobotB.score);

gotoxy(72,27);

cout <<" ";

//Xoa nhap vi tri

for(int i=75; i<101; i++){

gotoxy(i, 23);

cout << char(32);

Sleep(2);

}

for(int i=72; i<92; i++){

gotoxy(i, 25);

cout << char(32);

Sleep(2);

}

for(int i=72; i<92; i++){

gotoxy(i, 27);

cout << char(32);

Sleep(2);

}

moveIntersection1(RobotA, intersect1, numRows, numCols);

moveIntersection2(RobotB, intersect2, numRows, numCols);

calculateWinner(RobotA, RobotB);

//Vi tri cat nhau cua hai Robots

button:

while (true) {

if (\_kbhit()) {

char ch = \_getch();

if (ch == 27) {

return;

}

else if(ch == 'P' || ch == 'p'){

clrscr();

vitriCatNhau(RobotA, RobotB);

goto button;

}

else if(ch == '\r'){

clrscr();

ModeIntersection();

}

}

}

}

void moveWADS(){

vector<vector<int> > maze(numRows, vector<int>(numCols));

srand(time(0));

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

for (int j = 0; j < numCols; j++) {

maze[i][j] = rand() % 9 + 1;

}

}

//Doc ghi File de luu gia tri cua ma tran

ofstream outputFile("input.txt");

if (outputFile.is\_open()) {

outputFile << numRows << " " << numCols << endl;

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

for (int j = 0; j < numCols; j++) {

outputFile << maze[i][j] << "\t";

}

outputFile << endl;

}

outputFile.close();

}

//Khoi tao Me Cung luc dau

khoitaoBanCo(maze);

vientrongGame();

//Nhap vi tri 2 Robots

bool passed[numRows][numCols] = {false};

robot RobotA;

robot RobotB;

//Step ban dau 2 robots

RobotA.steps = 0;

gotoxy(93, 8);

TextColor(4);

cout << RobotA.steps;

RobotB.steps = 0;

gotoxy(93, 17);

TextColor(1);

cout << RobotB.steps;

//Vi tri bat dau cua Robots

PlaySound(TEXT("appear.wav"), NULL, SND\_FILENAME | SND\_ASYNC);

gotoxy(75, 23);

for(int i=0; i<15;i++){

TextColor(i);

gotoxy(75, 23);

cout << "Vi tri bat dau cua Robots";

Sleep(40);

hideCursor();

}

gotoxy(75, 23);

TextColor(11);

cout << "Vi tri bat dau cua Robots";

TextColor(7);

gotoxy(72,25);

cout << "->";

gotoxy(75, 25);

TextColor(4);

cout << "Robot A: ";

cin >> RobotA.x >> RobotA.y;

gotoxy(72,25);

cout <<" ";

RobotA.path.push\_back(maze[RobotA.x][RobotA.y]);

//Diem ban dau robot A

RobotA.score = maze[RobotA.x][RobotA.y];

gotoxy(87, 6);

TextColor(4);

cout << RobotA.score;

PlaySound(TEXT("tick.wav"), NULL, SND\_FILENAME | SND\_ASYNC);

veBanCo(maze, RobotA.x, RobotA.y, RobotB.x, RobotB.y, RobotA.score, RobotB.score);

TextColor(7);

gotoxy(72,27);

cout << "->";

gotoxy(75, 27);

TextColor(1);

cout << "Robot B: ";

cin >> RobotB.x >> RobotB.y;

gotoxy(72,27);

cout <<" ";

RobotB.path.push\_back(maze[RobotB.x][RobotB.y]);

//Diem ban dau robot B

RobotB.score = maze[RobotB.x][RobotB.y];

gotoxy(87, 15);

TextColor(1);

cout << RobotB.score;

PlaySound(TEXT("tick.wav"), NULL, SND\_FILENAME | SND\_ASYNC);

veBanCo(maze, RobotA.x, RobotA.y, RobotB.x, RobotB.y, RobotA.score, RobotB.score);

maze[RobotA.x][RobotA.y] = -1;

maze[RobotB.x][RobotB.y] = -2;

//Xoa nhap vi tri

for(int i=75; i<101; i++){

gotoxy(i, 23);

cout << char(32);

Sleep(2);

}

for(int i=72; i<92; i++){

gotoxy(i, 25);

cout << char(32);

Sleep(2);

}

for(int i=72; i<92; i++){

gotoxy(i, 27);

cout << char(32);

Sleep(2);

}

//Move WADS

int newX, newY;

while (true) {

char direction;

gotoxy(68, 25);

TextColor(4);

PlaySound(TEXT("go.wav"), NULL, SND\_FILENAME | SND\_ASYNC);

cout << "Enter direction for robot A (W/A/S/D): ";

cin >> direction;

gotoxy(66, 25);

for(int i=65; i<110; i++){

gotoxy(i, 25);

cout << char(32);

Sleep(10);

}

if (direction == 'W' || direction == 'w') {

newX = RobotA.x - 1;

newY = RobotA.y;

} else if (direction == 'A' || direction == 'a') {

newX = RobotA.x;

newY = RobotA.y - 1;

} else if (direction == 'S' || direction == 's') {

newX = RobotA.x + 1;

newY = RobotA.y;

} else if (direction == 'D' || direction == 'd') {

newX = RobotA.x;

newY = RobotA.y + 1;

} else {

gotoxy(67, 25);

TextColor(4);

PlaySound(TEXT("error.wav"), NULL, SND\_FILENAME | SND\_ASYNC);

cout << "Invalid direction. Please enter W/A/S/D." << endl;

Sleep(1000);

gotoxy(66, 25);

for(int i=65; i<110; i++){

gotoxy(i, 25);

cout << char(32);

Sleep(10);

}

continue;

}

bool canMove1 = false;

if (RobotA.x > 0 && isMoveValid(RobotA.x, RobotA.y, RobotA.x - 1, RobotA.y, maze, RobotA.x, RobotA.y, RobotB.x, RobotB.y)) {

canMove1 = true;

} else if (RobotA.y > 0 && isMoveValid(RobotA.x, RobotA.y, RobotA.x, RobotA.y - 1, maze, RobotA.y, RobotA.y, RobotB.x, RobotB.y)) {

canMove1 = true;

} else if (RobotA.x < maze.size() - 1 && isMoveValid(RobotA.x, RobotA.y, RobotA.x + 1, RobotA.y, maze, RobotA.x, RobotA.y, RobotB.x, RobotB.y)) {

canMove1 = true;

} else if (RobotA.y < maze[0].size() - 1 && isMoveValid(RobotA.x, RobotA.y, RobotA.x, RobotA.y + 1, maze, RobotA.x, RobotA.y, RobotB.x, RobotB.y)) {

canMove1 = true;

}

if (!canMove1) {

gotoxy(72, 25);

TextColor(4);

PlaySound(TEXT("error.wav"), NULL, SND\_FILENAME | SND\_ASYNC);

cout << "Robot A cannot move anymore!" << endl;

Sleep(1000);

gotoxy(66, 25);

for(int i=65; i<110; i++){

gotoxy(i, 25);

cout << char(32);

Sleep(10);

goto winner;

}

}

if (!isMoveValid(RobotA.x, RobotA.y, newX, newY, maze, RobotA.x, RobotA.y, RobotB.y, RobotB.y)) {

gotoxy(72, 25);

TextColor(10);

PlaySound(TEXT("error.wav"), NULL, SND\_FILENAME | SND\_ASYNC);

cout << "Robot A di chuyen khong hop le!" << endl;

Sleep(1000);

gotoxy(66, 25);

for(int i=65; i<110; i++){

gotoxy(i, 25);

cout << char(32);

Sleep(10);

}

continue;

}

RobotA.x = newX;

RobotA.y = newY;

RobotA.steps++;

gotoxy(93, 8);

TextColor(4);

cout << RobotA.steps;

RobotA.score += maze[newX][newY];

RobotA.path.push\_back(maze[newX][newY]);

if (maze[newX][newY] != -1) {

maze[newX][newY] = -1;

}

PlaySound(TEXT("move.wav"), NULL, SND\_FILENAME | SND\_ASYNC);

veBanCo(maze, RobotA.x, RobotA.y, RobotB.x, RobotB.y, RobotA.score, RobotB.score);

tryagain:

gotoxy(68, 25);

TextColor(1);

PlaySound(TEXT("go.wav"), NULL, SND\_FILENAME | SND\_ASYNC);

cout << "Enter direction for Robot 2 (I/J/L/K): ";

cin >> direction;

gotoxy(66, 25);

for(int i=65; i<110; i++){

gotoxy(i, 25);

cout << char(32);

Sleep(10);

}

if (direction == 'I' || direction == 'i') {

newX = RobotB.x - 1;

newY = RobotB.y;

} else if (direction == 'J' || direction == 'j') {

newX = RobotB.x;

newY = RobotB.y - 1;

} else if (direction == 'K' || direction == 'k') {

newX = RobotB.x + 1;

newY = RobotB.y;

} else if (direction == 'L' || direction == 'l') {

newX = RobotB.x;

newY = RobotB.y + 1;

} else {

gotoxy(67, 25);

TextColor(1);

PlaySound(TEXT("error.wav"), NULL, SND\_FILENAME | SND\_ASYNC);

cout << "Invalid direction. Please enter I/J/L/K." << endl;

Sleep(1000);

gotoxy(66, 25);

for(int i=65; i<110; i++){

gotoxy(i, 25);

cout << char(32);

Sleep(10);

}

goto tryagain;

}

bool canMove2 = false;

if (RobotB.x > 0 && isMoveValid(RobotB.x, RobotB.y, RobotB.x - 1, RobotB.y, maze, RobotA.x, RobotA.y, RobotB.x, RobotB.y)) {

canMove2 = true;

} else if (RobotB.y > 0 && isMoveValid(RobotB.x, RobotB.y, RobotB.x, RobotB.y - 1, maze, RobotA.y, RobotA.y, RobotB.x, RobotB.y)) {

canMove2 = true;

} else if (RobotB.x < maze.size() - 1 && isMoveValid(RobotB.x, RobotB.y, RobotB.x + 1, RobotB.y, maze, RobotA.x, RobotA.y, RobotB.x, RobotB.y)) {

canMove2 = true;

} else if (RobotB.y < maze[0].size() - 1 && isMoveValid(RobotB.x, RobotB.y, RobotB.x, RobotB.y + 1, maze, RobotA.x, RobotA.y, RobotB.x, RobotB.y)) {

canMove2 = true;

}

if (!canMove2) {

gotoxy(72, 25);

TextColor(1);

PlaySound(TEXT("error.wav"), NULL, SND\_FILENAME | SND\_ASYNC);

cout << "Robot B cannot move anymore";

Sleep(1000);

gotoxy(66, 25);

for(int i=65; i<110; i++){

gotoxy(i, 25);

cout << char(32);

Sleep(10);

}

goto winner;

}

if (!isMoveValid(RobotB.x, RobotB.y, newX, newY, maze, RobotA.x, RobotA.y, RobotB.y, RobotB.y)) {

gotoxy(72, 25);

TextColor(10);

PlaySound(TEXT("error.wav"), NULL, SND\_FILENAME | SND\_ASYNC);

cout << "Robot B di chuyen khong hop le!" << endl;

Sleep(1000);

gotoxy(66, 25);

for(int i=65; i<110; i++){

gotoxy(i, 25);

cout << char(32);

Sleep(10);

}

goto tryagain;

}

RobotB.x = newX;

RobotB.y = newY;

RobotB.steps++;

gotoxy(93, 17);

TextColor(1);

cout << RobotB.steps;

RobotB.score += maze[newX][newY];

RobotB.path.push\_back(maze[newX][newY]);

if (maze[newX][newY] != -2) {

maze[newX][newY] = -2;

}

PlaySound(TEXT("move.wav"), NULL, SND\_FILENAME | SND\_ASYNC);

veBanCo(maze, RobotA.x, RobotA.y, RobotB.x, RobotB.y, RobotA.score, RobotB.score);

}

winner:

calculateWinner(RobotA, RobotB);

veBanCo(maze, RobotA.x, RobotA.y, RobotB.x, RobotB.y, RobotA.score, RobotB.score);

button:

while (true) {

if (\_kbhit()) {

char ch = \_getch();

if (ch == 27) {

return;

}

else if(ch == '\r'){

clrscr();

moveWADS();

}

}

}

}

void normalMode(){

vector<vector<int> > maze(numRows, vector<int>(numCols));

srand(time(0));

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

for (int j = 0; j < numCols; j++) {

maze[i][j] = rand() % 9 + 1;

}

}

//Doc ghi File de luu gia tri cua ma tran

ofstream outputFile("input.txt");

if (outputFile.is\_open()) {

outputFile << numRows << " " << numCols << endl;

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

for (int j = 0; j < numCols; j++) {

outputFile << maze[i][j] << "\t";

}

outputFile << endl;

}

outputFile.close();

}

//Khoi tao Me Cung luc dau

khoitaoBanCo(maze);

vientrongGame();

//Nhap vi tri 2 Robots

bool passed[numRows][numCols] = {false};

robot RobotA;

robot RobotB;

//Step ban dau 2 robots

RobotA.steps = 0;

gotoxy(93, 8);

TextColor(4);

cout << RobotA.steps;

RobotB.steps = 0;

gotoxy(93, 17);

TextColor(1);

cout << RobotB.steps;

//Vi tri bat dau cua Robots

PlaySound(TEXT("appear.wav"), NULL, SND\_FILENAME | SND\_ASYNC);

gotoxy(75, 23);

for(int i=0; i<15;i++){

TextColor(i);

gotoxy(75, 23);

cout << "Vi tri bat dau cua Robots";

Sleep(60);

hideCursor();

}

gotoxy(75, 23);

TextColor(11);

cout << "Vi tri bat dau cua Robots";

TextColor(7);

gotoxy(72,25);

cout << "->";

gotoxy(75, 25);

TextColor(4);

cout << "Robot A: ";

cin >> RobotA.x >> RobotA.y;

gotoxy(72,25);

cout <<" ";

RobotA.score = maze[RobotA.x][RobotA.y];

gotoxy(87, 6);

TextColor(4);

cout << RobotA.score;

RobotA.path.push\_back(maze[RobotA.x][RobotA.y]);

Sleep(10);

PlaySound(TEXT("tick.wav"), NULL, SND\_FILENAME | SND\_ASYNC);

veBanCo(maze, RobotA.x, RobotA.y, RobotB.x, RobotB.y, RobotA.score, RobotB.score);

TextColor(7);

gotoxy(72,27);

cout << "->";

gotoxy(75, 27);

TextColor(1);

cout << "Robot B: ";

cin >> RobotB.x >> RobotB.y;

gotoxy(72,27);

cout <<" ";

RobotB.score = maze[RobotB.x][RobotB.y];

gotoxy(87, 15);

TextColor(1);

cout << RobotB.score;

RobotB.path.push\_back(maze[RobotB.x][RobotB.y]);

Sleep(10);

PlaySound(TEXT("tick.wav"), NULL, SND\_FILENAME | SND\_ASYNC);

veBanCo(maze, RobotA.x, RobotA.y, RobotB.x, RobotB.y, RobotA.score, RobotB.score);

maze[RobotA.x][RobotA.y] = -1;

maze[RobotB.x][RobotB.y] = -2;

//Xoa nhap vi tri

for(int i=75; i<101; i++){

gotoxy(i, 23);

cout << char(32);

Sleep(2);

}

for(int i=72; i<92; i++){

gotoxy(i, 25);

cout << char(32);

Sleep(2);

}

for(int i=72; i<92; i++){

gotoxy(i, 27);

cout << char(32);

Sleep(2);

}

veBanCo(maze, RobotA.x, RobotA.y, RobotB.x, RobotB.y, RobotA.score, RobotB.score);

move(RobotA, RobotB, maze, numRows, numCols);

while (true) {

if (\_kbhit()) {

char ch = \_getch();

if (ch == 27) {

return;

}

else if(ch == '\r'){

clrscr();

normalMode();

}

}

}

}

void chooseModeAuto(){

char ask;

int choice[] = {4,7};

int counter = 1;

char key;

gotoxy(44, 12);

TextColor(10);

cout << "Chon che do choi";

for(int i=0;;){

gotoxy(44,14);

TextColor(choice[0]);

cout<<"Normal";

gotoxy(44,16);

TextColor(choice[1]);

cout<<"Intersection";

key = \_getch();

if(key == 72 && (counter >=2 && counter <= 5))

{

PlaySound(TEXT("go.wav"), NULL, SND\_FILENAME | SND\_ASYNC);

counter--;

}

if(key == 80 && (counter >=1 && counter <= 4))

{

PlaySound(TEXT("go.wav"), NULL, SND\_FILENAME | SND\_ASYNC);

counter++;

}

if(key == '\r')

{

if(counter == 2)

{

PlaySound(TEXT("tick.wav"), NULL, SND\_FILENAME | SND\_ASYNC);

clrscr();

ModeIntersection();

clrscr();

return;

}

if(counter == 1)

{

PlaySound(TEXT("tick.wav"), NULL, SND\_FILENAME | SND\_ASYNC);

clrscr();

normalMode();

return;

}

}

choice[0] = 7;

choice[1] = 7;

if(counter == 1)

{

choice[0] = 4;

}

if(counter == 2)

{

choice[1] = 4;

}

}

}

void help(){

//Khung luat choi

gotoxy(0, 0);

TextColor(11);

cout << char(201);

for(int i=1; i<59; i++){

gotoxy(i, 0);

cout << char(205);

Sleep(1);

}

gotoxy(59,0);

cout << char(187);

for(int i = 1 ; i < 29; i++){

gotoxy(59, i);

cout << char(186);

Sleep(1);

}

gotoxy(59, 29); cout << char(188);

for(int i = 58; i >= 1; i--){

gotoxy(i, 29);

cout << char(205);

Sleep(1);

}

gotoxy(0, 29); cout << char(200);

for(int i = 28; i >= 1; i--){

gotoxy(0, i);

cout << char(186);

Sleep(1);

}

//Noi dung luat choi

gotoxy(13, 2);

for(int i=1; i<15; i++){

TextColor(i);

gotoxy(13, 2);

cout << " \_\_\_\_ \_ \_ \_ \_\_\_\_\_ \_\_\_\_ " ; gotoxy(13, 3);

cout << " | \_ \\| | | | | | \_\_\_\_/ \_\_\_| " ; gotoxy(13, 4);

cout << " | |\_) | | | | | | \_| \\\_\_\_ \\ " ; gotoxy(13, 5);

cout << " | \_ <| |\_| | |\_\_\_| |\_\_\_ \_\_\_) |" ; gotoxy(13, 6);

cout << " |\_| \\\_\\\\\_\_\_/|\_\_\_\_\_|\_\_\_\_\_|\_\_\_\_/ ";

Sleep(30);

}

PlaySound(TEXT("appear.wav"), NULL, SND\_FILENAME | SND\_ASYNC);

TextColor(12);

gotoxy(13, 2);

cout << " \_\_\_\_ \_ \_ \_ \_\_\_\_\_ \_\_\_\_ " ; gotoxy(13, 3);

cout << " | \_ \\| | | | | | \_\_\_\_/ \_\_\_| " ; gotoxy(13, 4);

cout << " | |\_) | | | | | | \_| \\\_\_\_ \\ " ; gotoxy(13, 5);

cout << " | \_ <| |\_| | |\_\_\_| |\_\_\_ \_\_\_) |" ; gotoxy(13, 6);

cout << " |\_| \\\_\\\\\_\_\_/|\_\_\_\_\_|\_\_\_\_\_|\_\_\_\_/ ";

TextColor(14);

gotoxy(6, 8);

cout << "+ Robots di theo 4 huong (tren, duoi, trai, phai)";

gotoxy(6, 10);

cout << "+ Robots se di theo huong co gia tri lon nhat";

gotoxy(6, 12); cout << "+ Voi mode Normal, Robots se tu dong di chuyen";

gotoxy(6, 13); cout << " theo huong ma hai nguoi choi nhap vao ban dau";

gotoxy(6, 14); cout << " robot thang khi co diem so lon nhat, khong duoc";

gotoxy(6, 15); cout << " di nhung o trung nhau";

gotoxy(6, 17); cout << "+ Voi mode Intersection, Robots se tu dong di";

gotoxy(6, 18); cout << " theo huong ma hai nguoi choi nhap vao ban dau";

gotoxy(6, 19); cout << " robot thang khi co diem so lon nhat, duoc phep";

gotoxy(6, 20); cout << " di nhung o trung nhau";

gotoxy(6, 22); cout << "+ Voi mode Adjust PvP, Robots di chuyen tuy y";

gotoxy(6, 23); cout << " theo huong ma nguoi choi muon di chuyen toi";

gotoxy(6, 24); cout << " robot thang khi co diem so lon nhat, khong duoc";

gotoxy(6, 25); cout << " di chuyen nhung o trung nhau";

gotoxy(6, 27); cout << "+ Khong duoc phep di chuyen ra khoi me cung";

//Khung dieu khien WADS

gotoxy(61, 0);

TextColor(10);

cout << char(201);

for(int i=62; i<119; i++){

gotoxy(i, 0);

cout << char(205);

Sleep(1);

}

gotoxy(119,0);

cout << char(187);

for(int i = 1 ; i < 14; i++){

gotoxy(119, i);

cout << char(186);

Sleep(1);

}

gotoxy(119, 14); cout << char(188);

for(int i = 118; i >= 61; i--){

gotoxy(i, 14);

cout << char(205);

Sleep(1);

}

gotoxy(61, 14); cout << char(200);

for(int i = 13; i >= 1; i--){

gotoxy(61, i);

cout << char(186);

Sleep(1);

}

//Noi dung khung Adjust PvP Mode

for(int i=1; i<15; i++){

TextColor(i);

gotoxy(82, 3);

cout << "Adjust PvP Mode";

Sleep(10);

}

PlaySound(TEXT("go.wav"), NULL, SND\_FILENAME | SND\_ASYNC);

TextColor(12);

gotoxy(82, 3);

cout << "Adjust PvP Mode";

TextColor(14);

gotoxy(78, 5); cout << "W/I: di chuyen len tren";

gotoxy(78, 7); cout << "A/J: di chuyen sang trai";

gotoxy(78, 9); cout << "D/L: di chuyen sang phai";

gotoxy(78, 11); cout << "S/K: di chuyen xuong duoi";

//Khung phim tat

gotoxy(61, 15);

TextColor(13);

cout << char(201);

for(int i=62; i<119; i++){

gotoxy(i, 15);

cout << char(205);

Sleep(1);

}

gotoxy(119,15);

cout << char(187);

for(int i = 16 ; i < 29; i++){

gotoxy(119, i);

cout << char(186);

Sleep(1);

}

gotoxy(119, 29); cout << char(188);

for(int i = 118; i >= 61; i--){

gotoxy(i, 29);

cout << char(205);

Sleep(1);

}

gotoxy(61, 29); cout << char(200);

for(int i = 28; i >= 16; i--){

gotoxy(61, i);

cout << char(186);

Sleep(1);

}

//Noi dung o phim tat

for(int i=1; i<15; i++){

TextColor(i);

gotoxy(82, 18);

cout << "Phim tat cho Game";

Sleep(10);

}

PlaySound(TEXT("go.wav"), NULL, SND\_FILENAME | SND\_ASYNC);

TextColor(12);

gotoxy(82, 18);

cout << "Phim tat cho Game";

TextColor(14);

gotoxy(74, 20); cout << "+ ESC: thoat man hinh hien tai";

gotoxy(74, 22); cout << "+ Enter: tiep tuc choi van game moi";

gotoxy(74, 24); cout << "+ Mode Intersection, nhan P de in ra";

gotoxy(74, 25); cout << " diem cat nhau Robots va so diem cat";

exitESC();

}

void aboutme(){

//Hai duong trang

for(int i = 41; i<80; i++){

TextColor(1);

gotoxy(i, 10);

cout << char(196);

Sleep(1);

}

for(int i = 41; i<80; i++){

TextColor(4);

gotoxy(i, 21);

cout << char(196);

Sleep(1);

}

PlaySound(TEXT("appear.wav"), NULL, SND\_FILENAME | SND\_ASYNC);

for(int i=1; i<15; i++){

TextColor(i);

gotoxy(46, 12);

cout << "Made by: Quach Phu Thuan";

Sleep(4);

}

for(int i=1; i<15; i++){

TextColor(i);

gotoxy(46, 14);

cout << "MSSV: 2251120446";

Sleep(4);

}

gotoxy(46, 16);

cout << "Class: CN22CLCE";

for(int i=1; i<15; i++){

TextColor(i);

gotoxy(46, 16);

cout << "Class: CN22CLCE";

Sleep(4);

}

for(int i=1; i<15; i++){

TextColor(i);

gotoxy(46, 18);

cout << "University Of Transport HCM City";

Sleep(4);

}

}

int main(){

hideCursor();

khunggiaodien();

tenGame();

int Set[] = {4,7,7,7,7}; // DEFAULT COLORS

int counter = 1;

char key;

for(int i=0;;)

{

gotoxy(50,15);

TextColor(Set[0]);

cout<<"Play Auto PvP";

gotoxy(50,16);

TextColor(Set[1]);

cout<<"Play Adjust PvP";

gotoxy(50,17);

TextColor(Set[2]);

cout<<"Help";

gotoxy(50,18);

TextColor(Set[3]);

cout<<"About me";

gotoxy(50,19);

TextColor(Set[4]);

cout<<"Exit";

key = \_getch();

if(key == 72 && (counter >=2 && counter <= 5))

{

PlaySound(TEXT("go.wav"), NULL, SND\_FILENAME | SND\_ASYNC);

counter--;

}

if(key == 80 && (counter >=1 && counter <= 4))

{

PlaySound(TEXT("go.wav"), NULL, SND\_FILENAME | SND\_ASYNC);

counter++;

}

if(key == '\r')

{

if(counter == 1)

{

PlaySound(TEXT("tick.wav"), NULL, SND\_FILENAME | SND\_ASYNC);

clrscr();

chooseModeAuto();

clrscr();

khunggiaodien();

tenGame();

continue;

}

if(counter == 2)

{

PlaySound(TEXT("tick.wav"), NULL, SND\_FILENAME | SND\_ASYNC);

clrscr();

moveWADS();

clrscr();

khunggiaodien();

tenGame();

continue;

}

if(counter == 3)

{

PlaySound(TEXT("tick.wav"), NULL, SND\_FILENAME | SND\_ASYNC);

clrscr();

help();

clrscr();

khunggiaodien();

tenGame();

continue;

}

if(counter == 4)

{

PlaySound(TEXT("tick.wav"), NULL, SND\_FILENAME | SND\_ASYNC);

clrscr();

aboutme();

exitESC();

clrscr();

khunggiaodien();

tenGame();

continue;

}

if(counter == 5)

{

PlaySound(TEXT("tick.wav"), NULL, SND\_FILENAME | SND\_ASYNC);

clrscr();

exitGame();

khunggiaodien();

tenGame();

continue;

}

}

Set[0] = 7;

Set[1] = 7;

Set[2] = 7;

Set[3] = 7;

Set[4] = 7;

if(counter == 1)

{

Set[0] = 4;

}

if(counter == 2)

{

Set[1] = 4;

}

if(counter == 3)

{

Set[2] = 4;

}

if(counter == 4)

{

Set[3] = 4;

}

if(counter == 5)

{

Set[4] = 4;

}

}

}