#include<iostream>

#include<windows.h>

#include <mmsystem.h>

using namespace std;

int X = 0;

int Y = 0;

const int s = 12;

int game[s][s] = {};

int live = 3;

bool hasStoped = false;

int score = 0;

void SetFruits();

int level = 1;

enum code { FRUIT = 2, MUSHROOM = 9, SPACE = 0, WALL = 5, PLAYER = 1 };

void ShowGame() {

Sleep(30);

system("cls");

cout << " Level " << level;

cout << endl;

for (size\_t i = 0; i < s; i++)

{

for (size\_t k = 0; k < s; k++)

{

if (i == 0 || i == s - 1) {

cout << "- ";

game[i][k] = WALL;

}

else if (k == 0) {

cout << "| ";

game[i][k] = WALL;

}

else if (k == s - 1) {

game[i][k] = WALL;

cout << "|";

}

else if (game[i][k] == WALL) {

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

SetConsoleTextAttribute(consolehwnd, FOREGROUND\_RED);

cout << (char)209 << " ";

SetConsoleTextAttribute(consolehwnd, FOREGROUND\_INTENSITY);

}

else if (game[i][k] == FRUIT) {

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

SetConsoleTextAttribute(consolehwnd, FOREGROUND\_GREEN);

cout << 'O' << " ";

SetConsoleTextAttribute(consolehwnd, FOREGROUND\_INTENSITY);

}

else if (game[i][k] == PLAYER) {

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

SetConsoleTextAttribute(consolehwnd, FOREGROUND\_BLUE);

cout << (char)2 << " ";

SetConsoleTextAttribute(consolehwnd, FOREGROUND\_INTENSITY);

}

else if (game[i][k] == MUSHROOM) {

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

SetConsoleTextAttribute(consolehwnd, FOREGROUND\_INTENSITY);

cout << (char)234 << " ";

SetConsoleTextAttribute(consolehwnd, FOREGROUND\_INTENSITY);

}

else if (game[i][k] == SPACE) {

cout << " ";

}

}

cout << endl;

}

cout << " SCORE : " << score;

/\*for (int i = 0; i < live; i++)

{

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

SetConsoleTextAttribute(consolehwnd, FOREGROUND\_RED);

cout << (char)3 << " ";

SetConsoleTextAttribute(consolehwnd, FOREGROUND\_INTENSITY);

}\*/

}

void ResetMushrooms() {

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

{

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

{

if (game[i][k] == MUSHROOM) {

game[i][k] = SPACE;

}

}

}

}

void SetWalls1()

{

game[3][3] = WALL;

game[4][3] = WALL;

game[5][3] = WALL;

}

void SetWalls2()

{

game[8][4] = WALL;

game[8][5] = WALL;

game[8][6] = WALL;

}

void SetWalls3()

{

game[3][7] = WALL;

game[4][7] = WALL;

game[5][7] = WALL;

}

bool isEmpty(int y, int x) {

if (x == 0 || y == 0 || x == s - 1 || y == s - 1) {

live--;

cout << "\a";

if (live == 0) {

hasStoped = true;

system("color 4");

system("cls");

Sleep(30);

cout << "GAME OVER!" << endl;

}

return false;

}

else if (game[y][x] == WALL) {

return false;

}

else if (game[y][x] == MUSHROOM) {

return false;

}

return true;

}

void GoToFood(int fY, int fX);

int FindFruitX();

int FindFruitY();

void IfMushroomReduceScore()

{

if (game[Y][X] == MUSHROOM)

{

score -= 50;

}

}

void StartLoop() {

while (true)

{

Sleep(30);

ShowGame();

if (hasStoped) {

return;

}

int fX = FindFruitX();

int fY = FindFruitY();

GoToFood(fY, fX);

if (score == 200)

{

system("color 2");

system("cls");

cout << "You Won!" << endl;

Sleep(4000);

break;

}

}

}

int FindFruitX()

{

for (int y = 0; y < s; y++)

{

for (int x = 0; x < s; x++)

{

if (game[y][x] == FRUIT)

return x;

}

}

}

int FindFruitY()

{

for (int y = 0; y < s; y++)

{

for (int x = 0; x < s; x++)

{

if (game[y][x] == FRUIT)

return y;

}

}

}

void GoUp();

void GoDown();

void GoRight();

void GoLeft();

void checkFruit()

{

int a1 = 0;

int a2 = 0;

int a3 = 0;

if (game[Y][X] == FRUIT) {

score += 10;

if (score % 50 == 0) {

ResetMushrooms();

}

if (score == 50) {

SetWalls1();

if (a1 == 0)

{

level++;

a1++;

}

}

else if (score == 30)

{

SetWalls2();

if (a2 == 0)

{

level++;

a2++;

}

}

else if (score == 150)

{

SetWalls3();

if (a3 == 0)

{

level++;

a3++;

}

}

SetFruits();

}

}

void GoUp()

{

if (game[Y - 1][X] != WALL)

{

game[Y][X] = SPACE;

Y--;

checkFruit();

IfMushroomReduceScore();

game[Y][X] = PLAYER;

ShowGame();

}

}

void GoDown()

{

if (game[Y + 1][X] != WALL)

{

game[Y][X] = SPACE;

Y++;

checkFruit();

IfMushroomReduceScore();

game[Y][X] = PLAYER;

ShowGame();

}

}

void GoRight()

{

if (game[Y][X + 1] != WALL)

{

game[Y][X] = SPACE;

X++;

checkFruit();

IfMushroomReduceScore();

game[Y][X] = PLAYER;

ShowGame();

}

else

{

if (game[Y - 1][X] != MUSHROOM && game[Y - 1][X] != WALL)

{

GoUp();

Sleep(250);

GoUp();

}

else if (game[Y + 1][X] != MUSHROOM && game[Y + 1][X] != WALL)

{

GoDown();

Sleep(250);

GoDown();

}

else

{

if (game[Y][X + 1] != WALL)

{

GoRight();

}

}

}

}

void GoLeft()

{

if (game[Y][X - 1] != WALL)

{

game[Y][X] = SPACE;

X--;

checkFruit();

IfMushroomReduceScore();

game[Y][X] = PLAYER;

ShowGame();

}

else

{

if (game[Y - 1][X] != MUSHROOM && game[Y - 1][X] != WALL)

{

GoUp();

Sleep(250);

GoUp();

}

else if (game[Y + 1][X] != MUSHROOM && game[Y + 1][X] != WALL)

{

GoDown();

Sleep(250);

GoDown();

}

else

{

if (game[Y][X - 1] != WALL)

{

GoLeft();

}

}

}

}

bool isNot5And9(int Y, int X)

{

if (game[Y][X] == SPACE)

return true;

return false;

}

void avoidMushroomRight(int fX, int fY)

{

if (Y > fY)

{

if (isNot5And9(Y - 1, X) && isNot5And9(Y - 1, X + 1))

{

GoUp();

Sleep(30);

GoRight();

}

else if (isNot5And9(Y - 1, X) && isNot5And9(Y - 2, X) && isNot5And9(Y - 2, X + 1))

{

GoUp();

Sleep(30);

GoUp();

Sleep(30);

GoRight();

}

else if (isNot5And9(Y + 1, X) && isNot5And9(Y + 1, X + 1))

{

GoUp();

Sleep(30);

GoRight();

}

else

{

if (game[Y][X + 1] != WALL)

{

GoRight();

}

}

}

else if (Y < fY)

{

if (isNot5And9(Y + 1, X) && isNot5And9(Y + 1, X + 1))

{

GoDown();

Sleep(30);

GoRight();

}

else if (isNot5And9(Y + 1, X) && isNot5And9(Y + 2, X) && isNot5And9(Y + 2, X + 1))

{

GoDown();

Sleep(30);

GoDown();

Sleep(30);

GoRight();

}

else if (isNot5And9(Y - 1, X) && isNot5And9(Y - 1, X + 1))

{

GoUp();

Sleep(30);

GoRight();

}

else

{

if (game[Y][X + 1] != WALL)

{

GoRight();

}

}

}

else

{

int counterUp = 0;

int counterDown = 0;

if (game[Y - 1][X] != WALL && game[Y - 1][X + 1] != WALL && game[Y - 1][X + 2] != WALL && game[Y][X + 2] != WALL)

{

if (game[Y - 1][X] == SPACE)

counterUp++;

if (game[Y - 1][X + 1] == SPACE)

counterUp++;

if (game[Y - 1][X + 2] == SPACE)

counterUp++;

if (game[Y][X + 2] == SPACE)

counterUp++;

}

if (game[Y + 1][X] != WALL && game[Y + 1][X + 1] != WALL && game[Y + 1][X + 2] != WALL && game[Y][X + 2] != WALL)

{

if (game[Y + 1][X] == SPACE)

counterDown++;

if (game[Y + 1][X + 1] == SPACE)

counterDown++;

if (game[Y + 1][X + 2] == SPACE)

counterDown++;

if (game[Y][X + 2] == SPACE)

counterDown++;

}

if (counterUp > counterDown)

{

if (game[Y - 1][X] != WALL && game[Y - 1][X + 1] != WALL && game[Y - 1][X + 2] != WALL && game[Y][X + 2] != WALL)

{

GoUp();

Sleep(30);

GoRight();

Sleep(30);

GoRight();

Sleep(30);

GoDown();

Sleep(30);

}

}

else if (counterUp < counterDown)

{

if (game[Y + 1][X] != WALL && game[Y + 1][X + 1] != WALL && game[Y + 1][X + 2] != WALL && game[Y][X + 2] != WALL)

{

GoDown();

Sleep(30);

GoRight();

Sleep(30);

GoRight();

Sleep(30);

GoUp();

Sleep(30);

}

}

else

{

if (game[Y][X + 1] != WALL)

{

GoRight();

if (game[Y][X + 1] != WALL)

{

if (X != fX)

GoRight();

}

}

}

}

}

void avoidMushroomLeft(int fX, int fY)

{

if (Y > fY)

{

if (isNot5And9(Y - 1, X) && isNot5And9(Y - 1, X - 1))

{

GoUp();

Sleep(30);

GoLeft();

}

else if (isNot5And9(Y - 1, X) && isNot5And9(Y - 2, X) && isNot5And9(Y - 2, X - 1))

{

GoUp();

Sleep(30);

GoUp();

Sleep(30);

GoLeft();

}

else if (isNot5And9(Y + 1, X) && isNot5And9(Y + 1, X - 1))

{

GoDown();

Sleep(30);

GoLeft();

}

else

{

if (game[Y][X - 1] != WALL)

{

GoLeft();

}

}

}

else if (Y < fY)

{

if (isNot5And9(Y + 1, X) && isNot5And9(Y + 1, X - 1))

{

GoDown();

Sleep(30);

GoLeft();

}

else if (isNot5And9(Y + 1, X) && isNot5And9(Y + 2, X) && isNot5And9(Y + 2, X - 1))

{

GoDown();

Sleep(30);

GoDown();

Sleep(30);

GoLeft();

}

else if (isNot5And9(Y - 1, X) && isNot5And9(Y - 1, X - 1))

{

GoUp();

Sleep(30);

GoLeft();

}

else

{

if (game[Y][X - 1] != WALL)

{

GoLeft();

}

}

}

else

{

int counterUp = 0;

int counterDown = 0;

if (game[Y - 1][X] != WALL && game[Y - 1][X - 1] != WALL && game[Y - 1][X - 2] != WALL && game[Y][X - 2] != WALL)

{

if (game[Y - 1][X] == SPACE)

counterUp++;

if (game[Y - 1][X - 1] == SPACE)

counterUp++;

if (game[Y - 1][X - 2] == SPACE)

counterUp++;

if (game[Y][X - 2] == SPACE)

counterUp++;

}

if (game[Y + 1][X] != WALL && game[Y + 1][X - 1] != WALL && game[Y + 1][X - 2] != WALL && game[Y][X - 2] != WALL)

{

if (game[Y + 1][X] == SPACE)

counterDown++;

if (game[Y + 1][X - 1] == SPACE)

counterDown++;

if (game[Y + 1][X - 2] == SPACE)

counterDown++;

if (game[Y][X - 2] == SPACE)

counterDown++;

}

if (counterUp > counterDown)

{

if (game[Y - 1][X] != WALL && game[Y - 1][X - 1] != WALL && game[Y - 1][X - 2] != WALL && game[Y - 3][X] != WALL)

{

GoUp();

Sleep(30);

GoLeft();

Sleep(30);

GoLeft();

Sleep(30);

GoDown();

Sleep(30);

}

}

else if (counterUp < counterDown)

{

if (game[Y + 1][X] != WALL && game[Y + 1][X - 1] != WALL && game[Y + 1][X - 2] != WALL && game[Y + 3][X] != WALL)

{

GoDown();

Sleep(30);

GoLeft();

Sleep(30);

GoLeft();

Sleep(30);

GoUp();

Sleep(30);

}

}

else

{

if (game[Y][X - 1] != WALL)

{

GoLeft();

if (X != fX)

{

if (game[Y][X - 1] != WALL)

GoLeft();

}

}

}

}

}

void avoidMushroomUp(int fX, int fY)

{

if (X < fX)

{

if (isNot5And9(Y, X + 1) && isNot5And9(Y - 1, X + 1))

{

GoRight();

Sleep(30);

GoUp();

}

else

{

if (game[Y][X + 1] != WALL)

{

GoRight();

}

else

{

GoUp();

}

}

}

else if (X > fX)

{

if (isNot5And9(Y, X - 1) && isNot5And9(Y - 1, X - 1))

{

GoLeft();

Sleep(30);

GoUp();

}

else

{

if (game[Y][X - 1] != WALL)

{

GoLeft();

}

else

{

GoUp();

}

}

}

else

{

if (isNot5And9(Y, X + 1) && isNot5And9(Y - 1, X + 1) && isNot5And9(Y - 2, X + 1))

{

GoRight();

Sleep(30);

GoUp();

Sleep(30);

GoUp();

}

else if (isNot5And9(Y, X - 1) && isNot5And9(Y - 1, X - 1) && isNot5And9(Y - 2, X - 1))

{

GoLeft();

Sleep(30);

GoUp();

Sleep(30);

GoUp();

}

else

{

GoUp();

}

}

}

void avoidMushroomDown(int fX, int fY)

{

if (X < fX)

{

if (isNot5And9(Y, X + 1) && isNot5And9(Y + 1, X + 1))

{

GoRight();

Sleep(30);

GoDown();

}

else

{

if (game[Y][X + 1] != WALL)

{

GoRight();

}

else

{

GoUp();

}

}

}

else if (X > fX)

{

if (isNot5And9(Y, X - 1) && isNot5And9(Y + 1, X - 1))

{

GoLeft();

Sleep(30);

GoDown();

}

else

{

if (game[Y][X - 1] != WALL)

{

GoLeft();

}

else

{

GoUp();

}

}

}

else

{

if (isNot5And9(Y, X + 1) && isNot5And9(Y + 1, X + 1) && isNot5And9(Y + 2, X + 1))

{

GoRight();

Sleep(30);

GoDown();

Sleep(30);

GoDown();

}

else if (isNot5And9(Y, X - 1) && isNot5And9(Y + 1, X - 1) && isNot5And9(Y + 2, X - 1))

{

GoLeft();

Sleep(30);

GoDown();

Sleep(30);

GoDown();

}

else

{

GoDown();

}

}

}

void GoToFood(int fY, int fX)

{

// Going to Right

if (X < fX)

{

// Trying to avoid mushrooms while going to right

if (game[Y][X + 1] == MUSHROOM)

{

avoidMushroomRight(fX, fY);

}

else if (isEmpty(Y, X + 1))

GoRight();

else

{

if (isEmpty(Y + 1, X + 1) == false && isEmpty(Y + 2, X + 1) == false)

{

GoUp();

if (isEmpty(Y, X + 1))

{

GoRight();

}

else if (isEmpty(Y, X + 1) == false && isEmpty(Y - 1, X + 1) == false)

{

GoRight();

}

else if (isEmpty(Y, X + 1) == false && isEmpty(Y - 1, X + 1) == true)

{

GoUp();

Sleep(30);

GoRight();

}

}

else if (isEmpty(Y - 1, X + 1) == false && isEmpty(Y - 2, X + 1) == false)

{

GoDown();

if (isEmpty(Y, X + 1))

{

GoRight();

}

else if (isEmpty(Y, X + 1) == false && isEmpty(Y - 1, X + 1) == false)

{

GoRight();

}

else if (isEmpty(Y, X + 1) == false && isEmpty(Y - 1, X + 1) == true)

{

GoDown();

Sleep(30);

GoRight();

}

}

else if (isEmpty(Y - 1, X + 1) == false && isEmpty(Y + 1, X + 1) == false)

{

if (isEmpty(Y + 1, X))

{

GoDown();

GoDown();

if (isEmpty(Y, X + 1))

{

GoRight();

}

else if (isEmpty(Y, X + 1) == false && isEmpty(Y - 1, X + 1) == false)

{

GoRight();

}

else if (isEmpty(Y, X + 1) == false && isEmpty(Y - 1, X + 1) == true)

{

GoDown();

Sleep(30);

GoRight();

}

}

else if (isEmpty(Y - 1, X))

{

GoUp();

GoUp();

if (isEmpty(Y, X + 1))

{

GoRight();

}

else if (isEmpty(Y, X + 1) == false && isEmpty(Y - 1, X + 1) == false)

{

GoRight();

}

else if (isEmpty(Y, X + 1) == false && isEmpty(Y - 1, X + 1) == true)

{

GoUp();

Sleep(30);

GoRight();

}

}

else

{

int counterUp = 0;

int counterDown = 0;

if (game[Y + 1][X] == SPACE)

counterDown++;

if (game[Y + 2][X] == SPACE)

counterDown++;

if (game[Y + 2][X + 1] == SPACE)

counterDown++;

if (game[Y + 2][X + 1] == SPACE)

counterDown++;

if (game[Y - 1][X] == SPACE)

counterUp++;

if (game[Y - 2][X] == SPACE)

counterUp++;

if (game[Y - 2][X + 1] == SPACE)

counterUp++;

if (game[Y - 2][X + 1] == SPACE)

counterUp++;

if (counterUp > counterDown)

{

GoUp();

GoUp();

if (isEmpty(Y, X + 1))

{

GoRight();

}

else if (isEmpty(Y, X + 1) == false && isEmpty(Y - 1, X + 1) == false)

{

GoRight();

}

else if (isEmpty(Y, X + 1) == false && isEmpty(Y - 1, X + 1) == true)

{

GoUp();

Sleep(30);

GoRight();

}

}

else if (counterUp < counterDown)

{

GoDown();

GoDown();

if (isEmpty(Y, X + 1))

{

GoRight();

}

else if (isEmpty(Y, X + 1) == false && isEmpty(Y - 1, X + 1) == false)

{

GoRight();

}

else if (isEmpty(Y, X + 1) == false && isEmpty(Y - 1, X + 1) == true)

{

GoDown();

Sleep(30);

GoRight();

}

}

else

{

GoDown();

GoDown();

if (isEmpty(Y, X + 1))

{

GoRight();

}

else if (isEmpty(Y, X + 1) == false && isEmpty(Y - 1, X + 1) == false)

{

GoRight();

}

else if (isEmpty(Y, X + 1) == false && isEmpty(Y - 1, X + 1) == true)

{

GoDown();

Sleep(30);

GoRight();

}

}

}

}

else if (game[Y][X + 1] == WALL && game[Y][X + 2] == WALL && game[Y][X + 3] == WALL)

{

int counterUp = 0;

int counterDown = 0;

if (game[Y - 1][X] == SPACE)

counterUp++;

if (game[Y - 1][X + 1] == SPACE)

counterUp++;

if (game[Y - 1][X + 2] == SPACE)

counterUp++;

if (game[Y - 1][X + 3] == SPACE)

counterUp++;

if (game[Y - 1][X + 4] == SPACE)

counterUp++;

if (game[Y + 1][X] == SPACE)

counterDown++;

if (game[Y + 1][X + 1] == SPACE)

counterDown++;

if (game[Y + 1][X + 2] == SPACE)

counterDown++;

if (game[Y + 1][X + 3] == SPACE)

counterDown++;

if (game[Y + 1][X + 4] == SPACE)

counterDown++;

if (counterUp > counterDown)

{

if (Y - 1 == fY)

{

GoUp();

Sleep(30);

GoRight();

}

else if (Y == fY)

{

GoUp();

Sleep(30);

GoRight();

}

else if (Y < fY)

{

GoDown();

Sleep(30);

GoRight();

if (X != fX)

{

Sleep(30);

GoRight();

}

}

else if (Y > fY)

{

GoUp();

Sleep(30);

GoRight();

if (X != fX)

{

Sleep(30);

GoRight();

}

}

}

else if (counterUp < counterDown)

{

if (Y + 1 == fY)

{

GoDown();

Sleep(30);

GoRight();

}

else if (Y == fY)

{

GoDown();

Sleep(30);

GoRight();

}

else if (Y > fY)

{

GoUp();

Sleep(30);

GoRight();

if (X != fX)

{

Sleep(30);

GoRight();

}

}

else if (Y < fY)

{

GoDown();

Sleep(30);

GoRight();

if (X != fX)

{

Sleep(30);

GoRight();

}

}

}

else

{

if (Y - 1 == fY)

{

GoUp();

Sleep(30);

GoRight();

}

else if (Y + 1 == fY)

{

GoDown();

Sleep(30);

GoRight();

}

else if (Y == fY)

{

GoUp();

Sleep(30);

GoRight();

}

else if (Y < fY)

{

GoDown();

Sleep(30);

GoRight();

if (X != fX)

{

Sleep(30);

GoRight();

}

}

else if (Y > fY)

{

GoUp();

Sleep(30);

GoRight();

if (X != fX)

{

Sleep(30);

GoRight();

}

}

}

}

}

}

// Going to Left

else if (X > fX)

{

// Trying to avoid mushrooms while going to left

if (game[Y][X - 1] == MUSHROOM)

{

avoidMushroomLeft(fX, fY);

}

else if (game[Y][X - 1] != WALL)

GoLeft();

else

{

if (isEmpty(Y + 1, X - 1) == false && isEmpty(Y + 2, X - 1) == false)

{

GoUp();

if (isEmpty(Y, X - 1))

{

GoLeft();

}

else if (isEmpty(Y, X - 1) == false && isEmpty(Y - 1, X - 1) == false)

{

GoLeft();

}

else if (isEmpty(Y, X - 1) == false && isEmpty(Y - 1, X - 1) == true)

{

GoUp();

Sleep(30);

GoLeft();

}

}

else if (isEmpty(Y - 1, X - 1) == false && isEmpty(Y - 2, X - 1) == false)

{

GoDown();

if (isEmpty(Y, X - 1))

{

GoLeft();

}

else if (isEmpty(Y, X - 1) == false && isEmpty(Y - 1, X - 1) == false)

{

GoLeft();

}

else if (isEmpty(Y, X - 1) == false && isEmpty(Y - 1, X - 1) == true)

{

GoDown();

Sleep(30);

GoLeft();

}

}

else if (isEmpty(Y - 1, X - 1) == false && isEmpty(Y + 1, X - 1) == false)

{

if (isEmpty(Y + 1, X))

{

GoDown();

Sleep(30);

GoDown();

if (isEmpty(Y, X - 1))

{

GoLeft();

}

else if (isEmpty(Y, X - 1) == false && isEmpty(Y - 1, X - 1) == false)

{

GoLeft();

}

else if (isEmpty(Y, X - 1) == false && isEmpty(Y - 1, X - 1) == true)

{

GoDown();

Sleep(30);

GoLeft();

}

}

else if (isEmpty(Y - 1, X))

{

GoUp();

Sleep(30);

GoUp();

if (isEmpty(Y, X - 1))

{

GoLeft();

}

else if (isEmpty(Y, X - 1) == false && isEmpty(Y - 1, X - 1) == false)

{

GoLeft();

}

else if (isEmpty(Y, X - 1) == false && isEmpty(Y - 1, X - 1) == true)

{

GoUp();

GoLeft();

}

}

else

{

int counterUp = 0;

int counterDown = 0;

if (game[Y + 1][X] == SPACE)

counterDown++;

if (game[Y + 2][X] == SPACE)

counterDown++;

if (game[Y + 2][X - 1] == SPACE)

counterDown++;

if (game[Y + 2][X - 1] == SPACE)

counterDown++;

if (game[Y - 1][X] == SPACE)

counterUp++;

if (game[Y - 2][X] == SPACE)

counterUp++;

if (game[Y - 2][X - 1] == SPACE)

counterUp++;

if (game[Y - 2][X - 1] == SPACE)

counterUp++;

if (counterUp > counterDown)

{

GoUp();

GoUp();

if (isEmpty(Y, X - 1))

{

GoLeft();

}

else if (isEmpty(Y, X - 1) == false && isEmpty(Y - 1, X - 1) == false)

{

GoLeft();

}

else if (isEmpty(Y, X - 1) == false && isEmpty(Y - 1, X - 1) == true)

{

GoUp();

GoLeft();

}

}

else if (counterUp < counterDown)

{

GoDown();

Sleep(30);

GoDown();

if (isEmpty(Y, X - 1))

{

GoLeft();

}

else if (isEmpty(Y, X - 1) == false && isEmpty(Y - 1, X - 1) == false)

{

GoLeft();

}

else if (isEmpty(Y, X - 1) == false && isEmpty(Y - 1, X - 1) == true)

{

GoDown();

GoLeft();

}

}

else

{

GoDown();

Sleep(30);

GoDown();

if (isEmpty(Y, X - 1))

{

GoLeft();

}

else if (isEmpty(Y, X - 1) == false && isEmpty(Y - 1, X - 1) == false)

{

GoLeft();

}

else if (isEmpty(Y, X - 1) == false && isEmpty(Y - 1, X - 1) == true)

{

GoDown();

GoLeft();

}

}

}

}

else if (game[Y][X - 1] == WALL && game[Y][X - 2] == WALL)

{

if (Y < fY)

{

GoDown();

Sleep(30);

GoDown();

Sleep(30);

GoDown();

Sleep(30);

}

else if (Y > fY)

{

GoUp();

Sleep(30);

GoUp();

Sleep(30);

GoUp();

Sleep(30);

}

}

}

}

// Going to Down

else if (Y < fY)

{

// Trying to avoid mushrooms while going to down

if (game[Y + 1][X] == MUSHROOM)

{

avoidMushroomDown(fX, fY);

}

else if (isEmpty(Y + 1, X))

GoDown();

else

{

if (game[Y + 1][X] == WALL && game[Y + 2][X] == WALL && game[Y + 3][X] == WALL)

{

if (X < fX)

{

GoRight();

}

else if (X > fX)

{

GoLeft();

}

else if (X == fX)

{

int counterRight = 0;

int counterLeft = 0;

if (game[Y][X + 1] == SPACE)

counterRight++;

if (game[Y + 1][X + 1] == SPACE)

counterRight++;

if (game[Y + 2][X + 1] == SPACE)

counterRight++;

if (game[Y + 3][X + 1] == SPACE)

counterRight++;

if (game[Y + 4][X + 1] == SPACE)

counterRight++;

if (game[Y][X - 1] == SPACE)

counterLeft++;

if (game[Y + 1][X - 1] == SPACE)

counterLeft++;

if (game[Y + 2][X - 1] == SPACE)

counterLeft++;

if (game[Y + 3][X - 1] == SPACE)

counterLeft++;

if (game[Y + 4][X - 1] == SPACE)

counterLeft++;

if (counterRight > counterLeft)

{

GoRight();

Sleep(30);

GoDown();

Sleep(30);

GoDown();

Sleep(30);

GoDown();

Sleep(30);

GoDown();

GoLeft();

}

else if (counterRight < counterLeft)

{

GoLeft();

Sleep(30);

GoDown();

Sleep(30);

GoDown();

Sleep(30);

GoDown();

Sleep(30);

GoDown();

GoRight();

}

else

{

GoLeft();

Sleep(30);

GoDown();

Sleep(30);

GoDown();

Sleep(30);

GoDown();

Sleep(30);

GoDown();

GoRight();

}

}

}

else if (game[Y + 1][X] == WALL && game[Y + 1][X + 1] == WALL && game[Y + 1][X + 2] == WALL)

{

if (game[Y][X - 1] == SPACE && game[Y + 1][X - 1] == SPACE && game[Y + 2][X - 1] == SPACE)

{

GoLeft();

Sleep(30);

GoDown();

Sleep(30);

GoDown();

}

else if (game[Y][X - 2] == SPACE && game[Y + 1][X - 2] == SPACE && game[Y + 2][X - 2] == SPACE)

{

GoLeft();

Sleep(30);

GoLeft();

Sleep(30);

GoDown();

Sleep(30);

GoDown();

}

else if (game[Y][X - 3] == SPACE && game[Y + 1][X - 3] == SPACE && game[Y + 2][X - 3] == SPACE)

{

GoLeft();

Sleep(30);

GoLeft();

Sleep(30);

GoLeft();

Sleep(30);

GoDown();

Sleep(30);

GoDown();

}

else

{

int counterC1 = 0;

int counterC2 = 0;

int counterC3 = 0;

if (game[Y][X - 1] == SPACE)

counterC3++;

if (game[Y + 1][X - 1] == SPACE)

counterC3++;

if (game[Y + 2][X - 1] == SPACE)

counterC3++;

if (game[Y][X - 2] == SPACE)

counterC2++;

if (game[Y + 1][X - 2] == SPACE)

counterC2++;

if (game[Y + 2][X - 2] == SPACE)

counterC2++;

if (game[Y][X - 3] == SPACE)

counterC1++;

if (game[Y + 1][X - 3] == SPACE)

counterC1++;

if (game[Y + 2][X - 3] == SPACE)

counterC1++;

if (counterC1 <= counterC2 && counterC1 <= counterC3)

{

GoLeft();

Sleep(30);

GoLeft();

Sleep(30);

GoLeft();

Sleep(30);

GoDown();

Sleep(30);

GoDown();

}

else if (counterC2 <= counterC3 && counterC2 <= counterC1)

{

GoLeft();

Sleep(30);

GoLeft();

Sleep(30);

GoDown();

Sleep(30);

GoDown();

}

else if (counterC3 <= counterC2 && counterC3 <= counterC1)

{

GoLeft();

Sleep(30);

GoDown();

Sleep(30);

GoDown();

}

else

{

GoLeft();

Sleep(30);

GoDown();

Sleep(30);

GoDown();

}

}

}

else if (game[Y + 1][X] == WALL && game[Y + 1][X + 1] == WALL && game[Y + 1][X - 1] == WALL)

{

if (X < fX && Y == fY)

{

GoRight();

Sleep(30);

GoRight();

Sleep(30);

GoDown();

}

else if (X < fX)

{

GoRight();

Sleep(30);

GoRight();

Sleep(30);

GoDown();

Sleep(30);

GoDown();

Sleep(30);

}

else if (X > fX && Y == fY)

{

GoLeft();

Sleep(30);

GoLeft();

Sleep(30);

GoDown();

}

else if (X > fX)

{

GoLeft();

Sleep(30);

GoLeft();

Sleep(30);

GoDown();

Sleep(30);

GoDown();

Sleep(30);

}

else if (X == fX)

{

if (game[Y][X - 2] == SPACE && game[Y + 1][X - 2] == SPACE && game[Y + 2][X - 2] == SPACE)

{

GoLeft();

Sleep(30);

GoLeft();

Sleep(30);

GoDown();

Sleep(30);

GoDown();

}

else if (game[Y][X - 3] == SPACE && game[Y + 1][X - 3] == SPACE && game[Y + 2][X - 3] == SPACE)

{

GoLeft();

Sleep(30);

GoLeft();

Sleep(30);

GoDown();

Sleep(30);

GoDown();

}

else if (game[Y][X - 4] == SPACE && game[Y + 1][X - 4] == SPACE && game[Y + 2][X - 4] == SPACE)

{

GoLeft();

Sleep(30);

GoLeft();

Sleep(30);

GoLeft();

Sleep(30);

GoDown();

Sleep(30);

GoDown();

}

else if (game[Y][X + 2] == SPACE && game[Y + 1][X + 2] == SPACE && game[Y + 2][X + 2] == SPACE)

{

GoRight();

Sleep(30);

GoRight();

Sleep(30);

GoDown();

Sleep(30);

GoDown();

}

else if (game[Y][X + 3] == SPACE && game[Y + 1][X + 3] == SPACE && game[Y + 2][X + 3] == SPACE)

{

GoRight();

Sleep(30);

GoRight();

Sleep(30);

GoRight();

Sleep(30);

GoDown();

Sleep(30);

GoDown();

}

else if (game[Y][X + 4] == SPACE && game[Y + 1][X + 4] == SPACE && game[Y + 2][X + 4] == SPACE)

{

GoRight();

Sleep(30);

GoRight();

Sleep(30);

GoRight();

Sleep(30);

GoRight();

Sleep(30);

GoDown();

Sleep(30);

GoDown();

}

else

{

int counterR = 0;

int counterL = 0;

if (game[Y][X - 1] == SPACE)

counterL++;

if (game[Y][X - 2] == SPACE)

counterL++;

if (game[Y + 1][X - 2] == SPACE)

counterL++;

if (game[Y + 2][X - 2] == SPACE)

counterL++;

if (game[Y][X + 1] == SPACE)

counterR++;

if (game[Y][X + 2] == SPACE)

counterR++;

if (game[Y + 1][X + 2] == SPACE)

counterR++;

if (game[Y + 2][X + 2] == SPACE)

counterR++;

if (counterL > counterR)

{

GoLeft();

Sleep(30);

GoLeft();

Sleep(30);

GoDown();

Sleep(30);

GoDown();

}

else if (counterL < counterR)

{

GoRight();

Sleep(30);

GoRight();

Sleep(30);

GoDown();

Sleep(30);

GoDown();

}

else

{

GoRight();

Sleep(30);

GoRight();

Sleep(30);

GoDown();

Sleep(30);

GoDown();

}

}

}

}

else if (game[Y + 1][X] == WALL && game[Y + 1][X - 1] == WALL && game[Y + 1][X - 2] == WALL)

{

if (game[Y][X + 1] == SPACE && game[Y + 1][X + 1] == SPACE && game[Y + 2][X + 1] == SPACE)

{

GoRight();

Sleep(30);

GoDown();

Sleep(30);

GoDown();

}

else if (game[Y][X + 2] == SPACE && game[Y + 1][X + 2] == SPACE && game[Y + 2][X + 2] == SPACE)

{

GoRight();

Sleep(30);

GoRight();

Sleep(30);

GoDown();

Sleep(30);

GoDown();

}

else if (game[Y][X + 3] == SPACE && game[Y + 1][X + 3] == SPACE && game[Y + 2][X + 3] == SPACE)

{

GoRight();

Sleep(30);

GoRight();

Sleep(30);

GoRight();

Sleep(30);

GoDown();

Sleep(30);

GoDown();

}

else

{

int counterC1 = 0;

int counterC2 = 0;

int counterC3 = 0;

int counterC4 = 0;

if (game[Y][X + 4] == SPACE)

counterC4++;

if (game[Y + 1][X + 4] == SPACE)

counterC4++;

if (game[Y + 2][X + 4] == SPACE)

counterC4++;

if (game[Y][X + 3] == SPACE)

counterC3++;

if (game[Y + 1][X + 3] == SPACE)

counterC3++;

if (game[Y + 2][X + 3] == SPACE)

counterC3++;

if (game[Y][X - 2] == SPACE)

counterC2++;

if (game[Y + 1][X + 2] == SPACE)

counterC2++;

if (game[Y + 2][X + 2] == SPACE)

counterC2++;

if (game[Y][X + 1] == SPACE)

counterC1++;

if (game[Y + 1][X + 1] == SPACE)

counterC1++;

if (game[Y + 2][X + 1] == SPACE)

counterC1++;

if (counterC1 <= counterC2 && counterC1 <= counterC3 && counterC1 <= counterC4)

{

GoRight();

Sleep(30);

GoDown();

Sleep(30);

GoDown();

}

else if (counterC2 <= counterC3 && counterC2 <= counterC1 && counterC2 <= counterC4)

{

GoRight();

Sleep(30);

GoRight();

Sleep(30);

GoDown();

Sleep(30);

GoDown();

}

else if (counterC3 <= counterC2 && counterC3 <= counterC1 && counterC3 <= counterC4)

{

GoRight();

Sleep(30);

GoRight();

Sleep(30);

GoRight();

Sleep(30);

GoDown();

Sleep(30);

GoDown();

}

else if (counterC4 <= counterC1 && counterC4 <= counterC2 && counterC4 <= counterC3)

{

GoRight();

Sleep(30);

GoRight();

Sleep(30);

GoRight();

Sleep(30);

GoRight();

Sleep(30);

GoDown();

Sleep(30);

GoDown();

}

else

{

GoRight();

Sleep(30);

GoDown();

Sleep(30);

GoDown();

}

}

}

}

}

// Going To Up

else if (Y > fY)

{

// Trying to avoid mushrooms while going to right

if (game[Y - 1][X] == MUSHROOM)

{

avoidMushroomUp(fX, fY);

}

else if (isEmpty(Y - 1, X))

GoUp();

else

{

if (game[Y - 1][X] == WALL && game[Y - 2][X] == WALL && game[Y - 3][X] == WALL)

{

if (X < fX)

{

GoRight();

}

else if (X > fX)

{

GoLeft();

}

else if (X == fX)

{

int counterRight = 0;

int counterLeft = 0;

if (game[Y][X + 1] == SPACE)

counterRight++;

if (game[Y - 1][X + 1] == SPACE)

counterRight++;

if (game[Y - 2][X + 1] == SPACE)

counterRight++;

if (game[Y - 3][X + 1] == SPACE)

counterRight++;

if (game[Y - 4][X + 1] == SPACE)

counterRight++;

if (game[Y][X - 1] == SPACE)

counterLeft++;

if (game[Y - 1][X - 1] == SPACE)

counterLeft++;

if (game[Y - 2][X - 1] == SPACE)

counterLeft++;

if (game[Y - 3][X - 1] == SPACE)

counterLeft++;

if (game[Y - 4][X - 1] == SPACE)

counterLeft++;

if (counterRight > counterLeft)

{

GoRight();

Sleep(30);

GoUp();

Sleep(30);

GoUp();

Sleep(30);

GoUp();

Sleep(30);

GoUp();

GoLeft();

}

else if (counterRight < counterLeft)

{

GoLeft();

Sleep(30);

GoUp();

Sleep(30);

GoUp();

Sleep(30);

GoUp();

Sleep(30);

GoUp();

GoRight();

}

else

{

GoLeft();

Sleep(30);

GoUp();

Sleep(30);

GoUp();

Sleep(30);

GoUp();

Sleep(30);

GoUp();

GoRight();

}

}

}

else if (game[Y - 1][X] == WALL && game[Y - 1][X + 1] == WALL && game[Y - 1][X + 2] == WALL)

{

if (game[Y][X - 1] == SPACE && game[Y - 1][X - 1] == SPACE && game[Y - 2][X - 1] == SPACE)

{

GoLeft();

Sleep(30);

GoUp();

Sleep(30);

GoUp();

}

else if (game[Y][X - 2] == SPACE && game[Y - 1][X - 2] == SPACE && game[Y - 2][X - 2] == SPACE)

{

GoLeft();

Sleep(30);

GoLeft();

Sleep(30);

GoUp();

Sleep(30);

GoUp();

}

else if (game[Y][X - 3] == SPACE && game[Y - 1][X - 3] == SPACE && game[Y - 2][X - 3] == SPACE)

{

GoLeft();

Sleep(30);

GoLeft();

Sleep(30);

GoLeft();

Sleep(30);

GoUp();

Sleep(30);

GoUp();

}

else

{

int counterC1 = 0;

int counterC2 = 0;

int counterC3 = 0;

if (game[Y][X - 1] == SPACE)

counterC3++;

if (game[Y - 1][X - 1] == SPACE)

counterC3++;

if (game[Y - 2][X - 1] == SPACE)

counterC3++;

if (game[Y][X - 2] == SPACE)

counterC2++;

if (game[Y - 1][X - 2] == SPACE)

counterC2++;

if (game[Y - 2][X - 2] == SPACE)

counterC2++;

if (game[Y][X - 3] == SPACE)

counterC1++;

if (game[Y - 1][X - 3] == SPACE)

counterC1++;

if (game[Y - 2][X - 3] == SPACE)

counterC1++;

if (counterC1 <= counterC2 && counterC1 <= counterC3)

{

GoLeft();

Sleep(30);

GoLeft();

Sleep(30);

GoLeft();

Sleep(30);

GoUp();

Sleep(30);

GoUp();

}

else if (counterC2 <= counterC3 && counterC2 <= counterC1)

{

GoLeft();

Sleep(30);

GoLeft();

Sleep(30);

GoUp();

Sleep(30);

GoUp();

}

else if (counterC3 <= counterC2 && counterC3 <= counterC1)

{

GoLeft();

Sleep(30);

GoUp();

Sleep(30);

GoUp();

}

else

{

GoLeft();

Sleep(30);

GoUp();

Sleep(30);

GoUp();

}

}

}

else if (game[Y - 1][X] == WALL && game[Y - 1][X + 1] == WALL && game[Y - 1][X - 1] == WALL)

{

if (X < fX && Y == fY)

{

GoRight();

Sleep(30);

GoRight();

Sleep(30);

GoUp();

}

else if (X < fX)

{

GoRight();

Sleep(30);

GoRight();

Sleep(30);

GoUp();

Sleep(30);

GoUp();

Sleep(30);

}

else if (X > fX && Y == fY)

{

GoLeft();

Sleep(30);

GoLeft();

Sleep(30);

GoUp();

}

else if (X > fX)

{

GoLeft();

Sleep(30);

GoLeft();

Sleep(30);

GoUp();

Sleep(30);

GoUp();

Sleep(30);

}

else if (X == fX)

{

if (game[Y][X - 2] == SPACE && game[Y + 1][X - 2] == SPACE && game[Y + 2][X - 2] == SPACE)

{

GoLeft();

Sleep(30);

GoLeft();

Sleep(30);

GoUp();

Sleep(30);

GoUp();

}

else if (game[Y][X - 3] == SPACE && game[Y + 1][X - 3] == SPACE && game[Y + 2][X - 3] == SPACE)

{

GoLeft();

Sleep(30);

GoLeft();

Sleep(30);

GoUp();

Sleep(30);

GoUp();

}

else if (game[Y][X - 4] == SPACE && game[Y + 1][X - 4] == SPACE && game[Y + 2][X - 4] == SPACE)

{

GoLeft();

Sleep(30);

GoLeft();

Sleep(30);

GoLeft();

Sleep(30);

GoUp();

Sleep(30);

GoUp();

}

else if (game[Y][X + 2] == SPACE && game[Y + 1][X + 2] == SPACE && game[Y + 2][X + 2] == SPACE)

{

GoRight();

Sleep(30);

GoRight();

Sleep(30);

GoUp();

Sleep(30);

GoUp();

}

else if (game[Y][X + 3] == SPACE && game[Y + 1][X + 3] == SPACE && game[Y + 2][X + 3] == SPACE)

{

GoRight();

Sleep(30);

GoRight();

Sleep(30);

GoRight();

Sleep(30);

GoUp();

Sleep(30);

GoUp();

}

else if (game[Y][X + 4] == SPACE && game[Y + 1][X + 4] == SPACE && game[Y + 2][X + 4] == SPACE)

{

GoRight();

Sleep(30);

GoRight();

Sleep(30);

GoRight();

Sleep(30);

GoRight();

Sleep(30);

GoUp();

Sleep(30);

GoUp();

}

else

{

int counterR = 0;

int counterL = 0;

if (game[Y][X - 1] == SPACE)

counterL++;

if (game[Y][X - 2] == SPACE)

counterL++;

if (game[Y + 1][X - 2] == SPACE)

counterL++;

if (game[Y + 2][X - 2] == SPACE)

counterL++;

if (game[Y][X + 1] == SPACE)

counterR++;

if (game[Y][X + 2] == SPACE)

counterR++;

if (game[Y + 1][X + 2] == SPACE)

counterR++;

if (game[Y + 2][X + 2] == SPACE)

counterR++;

if (counterL > counterR)

{

GoLeft();

Sleep(30);

GoLeft();

Sleep(30);

GoUp();

Sleep(30);

GoUp();

}

else if (counterL < counterR)

{

GoRight();

Sleep(30);

GoRight();

Sleep(30);

GoUp();

Sleep(30);

GoUp();

}

else

{

GoRight();

Sleep(30);

GoRight();

Sleep(30);

GoUp();

Sleep(30);

GoUp();

}

}

}

}

else if (game[Y - 1][X] == WALL && game[Y - 1][X - 1] == WALL && game[Y - 1][X - 2] == WALL)

{

if (game[Y][X + 1] == SPACE && game[Y - 1][X + 1] == SPACE && game[Y - 2][X + 1] == SPACE)

{

GoRight();

Sleep(30);

GoUp();

Sleep(30);

GoUp();

}

else if (game[Y][X + 2] == SPACE && game[Y - 1][X + 2] == SPACE && game[Y - 2][X + 2] == SPACE)

{

GoRight();

Sleep(30);

GoRight();

Sleep(30);

GoUp();

Sleep(30);

GoUp();

}

else if (game[Y][X + 3] == SPACE && game[Y - 1][X + 3] == SPACE && game[Y - 2][X + 3] == SPACE)

{

GoRight();

Sleep(30);

GoRight();

Sleep(30);

GoRight();

Sleep(30);

GoUp();

Sleep(30);

GoUp();

}

else

{

int counterC1 = 0;

int counterC2 = 0;

int counterC3 = 0;

int counterC4 = 0;

if (game[Y][X + 4] == SPACE)

counterC4++;

if (game[Y - 1][X + 4] == SPACE)

counterC4++;

if (game[Y - 2][X + 4] == SPACE)

counterC4++;

if (game[Y][X + 3] == SPACE)

counterC3++;

if (game[Y - 1][X + 3] == SPACE)

counterC3++;

if (game[Y - 2][X + 3] == SPACE)

counterC3++;

if (game[Y][X - 2] == SPACE)

counterC2++;

if (game[Y - 1][X + 2] == SPACE)

counterC2++;

if (game[Y - 2][X + 2] == SPACE)

counterC2++;

if (game[Y][X + 1] == SPACE)

counterC1++;

if (game[Y - 1][X + 1] == SPACE)

counterC1++;

if (game[Y - 2][X + 1] == SPACE)

counterC1++;

if (counterC1 <= counterC2 && counterC1 <= counterC3 && counterC1 <= counterC4)

{

GoRight();

Sleep(30);

GoUp();

Sleep(30);

GoUp();

}

else if (counterC2 <= counterC3 && counterC2 <= counterC1 && counterC2 <= counterC4)

{

GoRight();

Sleep(30);

GoRight();

Sleep(30);

GoUp();

Sleep(30);

GoUp();

}

else if (counterC3 <= counterC2 && counterC3 <= counterC1 && counterC3 <= counterC4)

{

GoRight();

Sleep(30);

GoRight();

Sleep(30);

GoRight();

Sleep(30);

GoUp();

Sleep(30);

GoUp();

}

else if (counterC4 <= counterC1 && counterC4 <= counterC2 && counterC4 <= counterC3)

{

GoRight();

Sleep(30);

GoRight();

Sleep(30);

GoRight();

Sleep(30);

GoRight();

Sleep(30);

GoUp();

Sleep(30);

GoUp();

}

else

{

GoRight();

Sleep(30);

GoUp();

Sleep(30);

GoUp();

}

}

}

}

}

}

void SetMushroom();

void SetFruits() {

while (true)

{

int rX = 1 + rand() % (9);

int rY = 1 + rand() % (9);

if (game[rY][rX] == SPACE) {

game[rY][rX] = FRUIT;

break;

}

}

SetMushroom();

SetMushroom();

}

void SetMushroom() {

while (true)

{

int rX = 1 + rand() % (9);

int rY = 1 + rand() % (9);

if (game[rY][rX] == SPACE) {

game[rY][rX] = MUSHROOM;

break;

}

}

}

void main() {

srand(time(0));

X = 1;

Y = 1;

game[Y][X] = PLAYER;

Sleep(500);

SetFruits();

ShowGame();

StartLoop();

}