Fast Mania



Session: 2022 – 2026

# Submitted by:

Your Name: MUHAMMAD FURQAN

Your Registration No: 2022R\2021-CS-199

# Supervised by:

Dr. Awais Hassan

Department of Computer Science

**University of Engineering and Technology Lahore Pakistan**

# Short Description and Story Writing of your Game:

# I have developed a Game inspiring from Snow Bros. In this game there is one main character and four enemies. Enemies have their own health and health decreases when the character’s bullet hit it and Character has its own and decreases when bullets fired on it. You Win is displayed when you win the game by eliminating all the enemies.

# Game Characters Description:

# There is one main character that can fire a bullet both right and left. The four enemies fire their bullets left and right respectively.

# Game Objects Description:

# There is maze in rectangle format. There are different trays there and character and enemy can move over there. There are stairs for the character through which it can move up to another tray to kill the enemy.

# Rules and Interactions:

# Move the arrow up key to move up. Move the arrow down key to move down. Move the arrow right key to move right. Move the arrow down key to move left. Press space key to fire bullet from right side. Press control key to fire bullet from left side. Only character can move up and down with stairs. Character can have only three healths. The score can be seen at the bottom(middle) of the screen.

# Goal of the Game:

# My goal is to develop a shooting game like snow bros. In future, I will further add the functionality that that there more enemies and more levels and some complicated levels. I will develop to make a bonus level and there will be a big enemy facing the character and it has more health.

# Wireframes of the Game

# Game Entrance

# 

# Game Menus

# 

# Menu 1

# 

# Menu 2

# 

# Menu 3

# 

# Function Protypes:

# void storeCoordinates();

# void LoadCoordinate();

# void storeScore();

# void storeHealth();

# HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

# void healthDecrement();

# // functions prototypes

# void printMaze();

# void gotoxy(int x, int y);

# char getcharatxy(short int x, short int y);

# void printCharacter();

# void eraseCharacter();

# void generateLift1();

# void generateLift2();

# void generateLift3();

# void generateLift4();

# void generateLift5();

# void generateLift6();

# void generateLift7();

# void moveMarioLeft();

# void moveMarioRight();

# void moveMarioUp();

# void moveMarioDown();

# void moveEnemy1();

# void moveEnemy2();

# void moveEnemy3();

# void moveEnemy4();

# void printEnemy1(int e1nemyX);

# void printEnemy2(int e2nemyX);

# void printEnemy3(int e3nemyX);

# void printEnemy4(int e4nemyY);

# void eraseEnemy1();

# void eraseEnemy2();

# void eraseEnemy3();

# void eraseEnemy4();

# void generateBulletRight();

# void moveBulletRight();

# void eraseBulletRight(int x, int y);

# void printBulletRight(int x, int y);

# void removeBulletfromArrayRight(int index);

# void bulletCollisionWithEnemyRight();

# void generateBulletLeft();

# void moveBulletLeft();

# void eraseBulletLeft(int x, int y);

# void printBulletLeft(int x, int y);

# void removeBulletfromArrayLeft(int index);

# void bulletCollisionWithEnemyLeft();

# void printScore();

# void healthDecrement();

# void addScore();

# // functions for generating bullet of enemies

# void generateBulletEnemy1Left();

# void generateBulletEnemy1Right();

# void generateBulletEnemy2Left();

# void generateBulletEnemy2Right();

# void generateBulletEnemy3Left();

# void generateBulletEnemy3Right();

# void generateBulletEnemy4Left();

# void generateBulletEnemy4Right();

# // BULLET Movement of Enemies to RIGHT

# // Enemy 1

# void moveBulletRightEnemy1();

# void printBulletRightEnemy1(int x, int y);

# void eraseBulletRightEnemy1(int x, int y);

# void removeBulletfromArrayRightEnemy1(int index);

# // Enemy2

# void moveBulletRightEnemy2();

# void printBulletRightEnemy2(int x, int y);

# void eraseBulletRightEnemy2(int x, int y);

# void removeBulletfromArrayRightEnemy2(int index);

# // Enemy3

# void moveBulletRightEnemy3();

# void printBulletRightEnemy3(int x, int y);

# void eraseBulletRightEnemy3(int x, int y);

# void removeBulletfromArrayRightEnemy3(int index);

# // Enemy4

# void moveBulletRightEnemy4();

# void printBulletRightEnemy4(int x, int y);

# void eraseBulletRightEnemy4(int x, int y);

# void removeBulletfromArrayRightEnemy4(int index);

# // BULLET Movement of Enemies to LEFT

# // Enemy 1

# void moveBulletLeftEnemy1();

# void printBulletLeftEnemy1(int x, int y);

# void eraseBulletLeftEnemy1(int x, int y);

# void removeBulletfromArrayLeftEnemy1(int index);

# // Enemy2

# void moveBulletLeftEnemy2();

# void printBulletLeftEnemy2(int x, int y);

# void eraseBulletLeftEnemy2(int x, int y);

# void removeBulletfromArrayLeftEnemy2(int index);

# // Enemy 3

# void moveBulletLeftEnemy3();

# void printBulletLeftEnemy3(int x, int y);

# void eraseBulletLeftEnemy3(int x, int y);

# void removeBulletfromArrayLeftEnemy3(int index);

# // Enemy 4

# void moveBulletLeftEnemy4();

# void printBulletLeftEnemy4(int x, int y);

# void eraseBulletLeftEnemy4(int x, int y);

# void removeBulletfromArrayLeftEnemy4(int index);

# void printStatus();

# void printHealth();

# void printGameName();

# void removeHealth();

# string loginOption;

# void gameEntrance();

# void topHeader();

# string loginMenu();

# void menu1();

# void menu2();

# void menu3();

# void getCharacter();

# void healthDecrement();

# void winningInterface();

# Complete Code:

# #include <iostream>

# #include <conio.h>

# #include <windows.h>

# #include <fstream>

# using namespace std;

# void storeCoordinates();

# void LoadCoordinate();

# void storeScore();

# void storeHealth();

# HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

# void healthDecrement();

# // functions prototypes

# void printMaze();

# void gotoxy(int x, int y);

# char getcharatxy(short int x, short int y);

# void printCharacter();

# void eraseCharacter();

# void generateLift1();

# void generateLift2();

# void generateLift3();

# void generateLift4();

# void generateLift5();

# void generateLift6();

# void generateLift7();

# void moveMarioLeft();

# void moveMarioRight();

# void moveMarioUp();

# void moveMarioDown();

# void moveEnemy1();

# void moveEnemy2();

# void moveEnemy3();

# void moveEnemy4();

# void printEnemy1(int e1nemyX);

# void printEnemy2(int e2nemyX);

# void printEnemy3(int e3nemyX);

# void printEnemy4(int e4nemyY);

# void eraseEnemy1();

# void eraseEnemy2();

# void eraseEnemy3();

# void eraseEnemy4();

# void generateBulletRight();

# void moveBulletRight();

# void eraseBulletRight(int x, int y);

# void printBulletRight(int x, int y);

# void removeBulletfromArrayRight(int index);

# void bulletCollisionWithEnemyRight();

# void generateBulletLeft();

# void moveBulletLeft();

# void eraseBulletLeft(int x, int y);

# void printBulletLeft(int x, int y);

# void removeBulletfromArrayLeft(int index);

# void bulletCollisionWithEnemyLeft();

# void printScore();

# void healthDecrement();

# void addScore();

# // functions for generating bullet of enemies

# void generateBulletEnemy1Left();

# void generateBulletEnemy1Right();

# void generateBulletEnemy2Left();

# void generateBulletEnemy2Right();

# void generateBulletEnemy3Left();

# void generateBulletEnemy3Right();

# void generateBulletEnemy4Left();

# void generateBulletEnemy4Right();

# // BULLET Movement of Enemies to RIGHT

# // Enemy 1

# void moveBulletRightEnemy1();

# void printBulletRightEnemy1(int x, int y);

# void eraseBulletRightEnemy1(int x, int y);

# void removeBulletfromArrayRightEnemy1(int index);

# // Enemy2

# void moveBulletRightEnemy2();

# void printBulletRightEnemy2(int x, int y);

# void eraseBulletRightEnemy2(int x, int y);

# void removeBulletfromArrayRightEnemy2(int index);

# // Enemy3

# void moveBulletRightEnemy3();

# void printBulletRightEnemy3(int x, int y);

# void eraseBulletRightEnemy3(int x, int y);

# void removeBulletfromArrayRightEnemy3(int index);

# // Enemy4

# void moveBulletRightEnemy4();

# void printBulletRightEnemy4(int x, int y);

# void eraseBulletRightEnemy4(int x, int y);

# void removeBulletfromArrayRightEnemy4(int index);

# // BULLET Movement of Enemies to LEFT

# // Enemy 1

# void moveBulletLeftEnemy1();

# void printBulletLeftEnemy1(int x, int y);

# void eraseBulletLeftEnemy1(int x, int y);

# void removeBulletfromArrayLeftEnemy1(int index);

# // Enemy2

# void moveBulletLeftEnemy2();

# void printBulletLeftEnemy2(int x, int y);

# void eraseBulletLeftEnemy2(int x, int y);

# void removeBulletfromArrayLeftEnemy2(int index);

# // Enemy 3

# void moveBulletLeftEnemy3();

# void printBulletLeftEnemy3(int x, int y);

# void eraseBulletLeftEnemy3(int x, int y);

# void removeBulletfromArrayLeftEnemy3(int index);

# // Enemy 4

# void moveBulletLeftEnemy4();

# void printBulletLeftEnemy4(int x, int y);

# void eraseBulletLeftEnemy4(int x, int y);

# void removeBulletfromArrayLeftEnemy4(int index);

# void printStatus();

# void printHealth();

# void printGameName();

# void removeHealth();

# string loginOption;

# void gameEntrance();

# void topHeader();

# string loginMenu();

# void menu1();

# void menu2();

# void menu3();

# void getCharacter();

# void healthDecrement();

# void winningInterface();

# // character variables

# int pX = 6;

# int pY = 32;

# int bulletRightX[100];

# int bulletRightY[100];

# int bulletCountRight = 0;

# int bulletLeftX[100];

# int bulletLeftY[100];

# int bulletCountLeft = 0;

# // BULLET variables for Enemies RIGHT

# // Enemy 1

# int enemy1BulletRightX[100];

# int enemy1BulletRightY[100];

# int bulletCountRightEnemy1 = 0;

# // Enemy 2

# int enemy2BulletRightX[100];

# int enemy2BulletRightY[100];

# int bulletCountRightEnemy2 = 0;

# // Enemy 3

# int enemy3BulletRightX[100];

# int enemy3BulletRightY[100];

# int bulletCountRightEnemy3 = 0;

# // Enemy 4

# int enemy4BulletRightX[100];

# int enemy4BulletRightY[100];

# int bulletCountRightEnemy4 = 0;

# // BULLET variables for Enemies LEFT

# // Enemy 1

# int enemy1BulletLeftX[100];

# int enemy1BulletLeftY[100];

# int bulletCountLeftEnemy1 = 0;

# // Enemy 2

# int enemy2BulletLeftX[100];

# int enemy2BulletLeftY[100];

# int bulletCountLeftEnemy2 = 0;

# // Enemy 3

# int enemy3BulletLeftX[100];

# int enemy3BulletLeftY[100];

# int bulletCountLeftEnemy3 = 0;

# // Enemy 4

# int enemy4BulletLeftX[100];

# int enemy4BulletLeftY[100];

# int bulletCountLeftEnemy4 = 0;

# int e1nemyX = 60;

# int e1nemyY = 5;

# int e2nemyX = 12;

# int e2nemyY = 19;

# int e3nemyX = 45;

# int e3nemyY = 26;

# int e4nemyX = 109;

# int e4nemyY = 19;

# // for the draw of maze

# char box1 = 219;

# char box2 = 177;

# char box3 = 178;

# char left\_character = 174;

# char right\_character = 175;

# char box = 219;

# char emoji = 2;

# char a = 219;

# char lift = 22;

# int healthCounter = 100;

# char enemyBulletName = 15;

# int bulletColour = 13;

# int timer = 0;

# string enemyDirection1 = "Left";

# string enemyDirection2 = "Left";

# string enemyDirection3 = "Left";

# string enemyDirection4 = "Left";

# int score = 0;

# string status = "Playing";

# string gameName = "FAST MANIA";

# // main character

# char character1[5] = {' ', ' ', emoji, ' ', ' '};

# char character2[5] = {left\_character, box, box, box, right\_character};

# char character3[5] = {'\_', box, ' ', box, '\_'};

# // enemy 1

# char e1nemy1[6] = {'<', '\_', box, box, '\_', '>'};

# char e1nemy2[6] = {' ', '0', ' ', ' ', '0', ' '};

# // enemy 2

# char e2nemy1[6] = {'<', '\_', box, box, '\_', '>'};

# char e2nemy2[6] = {' ', '0', ' ', ' ', '0', ' '};

# // enemy 3

# char e3nemy1[6] = {'<', '\_', box, box, '\_', '>'};

# char e3nemy2[6] = {' ', '0', ' ', ' ', '0', ' '};

# // enemy 4

# char e4nemy1[6] = {'<', '\_', box, box, '\_', '>'};

# char e4nemy2[6] = {' ', '0', ' ', ' ', '0', ' '};

# int enemy1Health = 50;

# int enemy2Health = 50;

# int enemy3Health = 50;

# int enemy4Health = 50;

# int winningCount = 0;

# // main function

# int main()

# {

# while (loginOption != "4")

# {

# if (winningCount == 4)

# {

# system("cls");

# topHeader();

# winningInterface();

# getCharacter();

# break;

# }

# system("cls");

# gameEntrance();

# getche();

# system("cls");

# topHeader();

# loginOption = loginMenu();

# system("cls");

# if (loginOption == "1")

# {

# LoadCoordinate();

# menu1();

# }

# else if (loginOption == "2")

# {

# menu2();

# }

# else if (loginOption == "3")

# {

# menu3();

# }

# else if (loginOption != "1" || loginOption != "2" || loginOption != "3" || loginOption != "4")

# {

# topHeader();

# cout << "\n\n\n\n";

# cout << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << endl;

# cout << "\n\n";

# cout << "Thanks for Playing Fast Mania\n";

# }

# storeCoordinates();

# storeHealth();

# storeScore();

# }

# }

# void getCharacter()

# {

# cout << "\n\n\n";

# cout << "Press any key to continue.......";

# getch();

# }

# // menu 1 of the game the main working of the game

# void menu1()

# {

# bool isGameRunning = true;

# system("cls");

# printMaze();

# printCharacter();

# LoadCoordinate();

# while (isGameRunning == true)

# {

# healthDecrement();

# printScore();

# if (GetAsyncKeyState(VK\_LEFT))

# {

# moveMarioLeft();

# }

# if (GetAsyncKeyState(VK\_RIGHT))

# {

# moveMarioRight();

# }

# if (GetAsyncKeyState(VK\_UP))

# {

# moveMarioUp();

# }

# if (GetAsyncKeyState(VK\_DOWN))

# {

# moveMarioDown();

# }

# if (GetAsyncKeyState(VK\_SPACE))

# {

# generateBulletRight();

# }

# if (GetAsyncKeyState(VK\_CONTROL))

# {

# generateBulletLeft();

# }

# if (GetAsyncKeyState(VK\_TAB))

# {

# isGameRunning = false;

# }

# printHealth();

# if (healthCounter < 0)

# {

# isGameRunning = false;

# }

# if (timer == 3)

# {

# if (enemy1Health > 0)

# {

# moveEnemy1();

# }

# else if (enemy1Health == 0)

# {

# eraseEnemy1();

# winningCount++;

# }

# if (enemy2Health > 0)

# {

# moveEnemy2();

# }

# else if (enemy2Health == 0)

# {

# eraseEnemy2();

# winningCount++;

# }

# if (enemy3Health > 0)

# {

# moveEnemy3();

# }

# else if (enemy3Health == 0)

# {

# eraseEnemy3();

# winningCount++;

# }

# if (enemy4Health > 0)

# {

# moveEnemy4();

# }

# else if (enemy4Health == 0)

# {

# eraseEnemy4();

# winningCount++;

# }

# timer = 0;

# }

# if (winningCount == 4)

# {

# break;

# }

# generateLift1();

# generateLift2();

# generateLift3();

# generateLift4();

# generateLift5();

# generateLift6();

# generateLift7();

# printStatus();

# printGameName();

# moveBulletRight();

# bulletCollisionWithEnemyRight();

# moveBulletLeft();

# bulletCollisionWithEnemyLeft();

# moveBulletRightEnemy1();

# moveBulletLeftEnemy1();

# moveBulletRightEnemy2();

# moveBulletLeftEnemy2();

# moveBulletRightEnemy3();

# moveBulletLeftEnemy3();

# moveBulletRightEnemy4();

# moveBulletLeftEnemy4();

# healthDecrement();

# timer++;

# Sleep(100);

# }

# if (healthCounter < 0)

# {

# topHeader();

# cout << "\n\n\n\n";

# cout << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << endl;

# cout << "\n\n";

# cout << "You Lost the Game!\n"

# << endl;

# getCharacter();

# getCharacter();

# }

# }

# void winningInterface()

# {

# cout << "\n\n\n\n";

# cout << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << endl;

# cout << "\n\n";

# cout << a << " " << a << " " << a << a << a << a << " " << a << " " << a << " " << a << " " << a << " " << a << " " << a << a << a << " " << a << a << a << " " << a << endl;

# cout << a << " " << a << " " << a << " " << a << " " << a << " " << a << " " << a << " " << a << " " << a << " "

# << " " << a << " " << a << " " << a << " " << a << endl;

# cout << a << a << a << a << a << " " << a << " " << a << " " << a << " " << a << " " << a << " " << a << " " << a << " "

# << " " << a << " " << a << " " << a << " " << a << endl;

# cout << " " << a << " "

# << " " << a << " " << a << " " << a << " " << a << " " << a << " " << a << " " << a << " "

# << " " << a << " " << a << " " << a << " " << a << endl;

# cout << " " << a << " "

# << " " << a << a << a << a << " " << a << a << a << a << a << " " << a << a << a << a << a << a << a << " " << a << a << a << " " << a << " " << a << a << a << endl;

# }

# // menu 2 of the game to show the instructions

# void menu2()

# {

# topHeader();

# cout << "\n\n\n\n";

# cout << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << endl;

# cout << "\n\n";

# cout << "1. Move the Arrow Up Key to move UP" << endl;

# cout << "2. Move the Arrow Down Key to move DOWN" << endl;

# cout << "3. Move the Arrow Right Key to move RIGHT" << endl;

# cout << "4. Move the Arrow Down Key to move LEFT" << endl;

# cout << "5. Press Space Key to fire Bullet from right side" << endl;

# cout << "6. Press Control Key to fire Bullet from left side" << endl;

# cout << "7. Character can only move UP and DOWN with Stairs" << endl;

# cout << "8. Character can have only 3 Healths" << endl;

# cout << "9. The Score can be seen at the bottom(middle) of the screen" << endl;

# getCharacter();

# }

# // Menu 3 of the game for Character Information

# void menu3()

# {

# topHeader();

# cout << "\n\n\n\n";

# cout << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << endl;

# cout << "\n\n";

# cout << "Main Character :\n\n";

# for (int i = 0; i < 5; i++)

# {

# cout << character1[i];

# }

# cout << endl;

# for (int i = 0; i < 5; i++)

# {

# cout << character2[i];

# }

# cout << endl;

# for (int i = 0; i < 5; i++)

# {

# cout << character3[i];

# }

# cout << "\n\n\n";

# cout << "Enemy # 1 :\n\n";

# for (int index = 0; index < 6; index++)

# {

# cout << e1nemy1[index];

# }

# cout << endl;

# for (int index = 0; index < 6; index++)

# {

# cout << e1nemy2[index];

# }

# cout << "\n\n\n";

# cout << "Enemy # 2 :\n\n";

# for (int index = 0; index < 6; index++)

# {

# cout << e2nemy1[index];

# }

# cout << endl;

# for (int index = 0; index < 6; index++)

# {

# cout << e2nemy2[index];

# }

# cout << "\n\n\n";

# cout << "Enemy # 3 :\n\n";

# for (int index = 0; index < 6; index++)

# {

# cout << e3nemy1[index];

# }

# cout << endl;

# for (int index = 0; index < 6; index++)

# {

# cout << e3nemy2[index];

# }

# cout << "\n\n\n";

# cout << "Enemy # 4 :\n\n";

# for (int index = 0; index < 6; index++)

# {

# cout << e4nemy1[index];

# }

# cout << endl;

# for (int index = 0; index < 6; index++)

# {

# cout << e4nemy2[index];

# }

# cout << endl;

# getCharacter();

# }

# // login menu

# string loginMenu()

# {

# string option;

# cout << "\n\n\n\n";

# cout << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << endl;

# cout << "\n\n";

# cout << "1. Start the game\n";

# cout << "2. Instructions\n";

# cout << "3. Character Information\n";

# cout << "4. Exit the game\n\n\n";

# cout << "Choose your Option...... ";

# cin >> option;

# return option;

# }

# // game Entrance

# void gameEntrance()

# {

# cout << " " << endl;

# cout << " " << endl;

# cout << " " << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << " " << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << " " << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << " " << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << endl;

# cout << " " << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << " " << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << " " << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << " " << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << endl;

# cout << " " << a << a << a << " " << a << a << a << " " << a << a << a << " " << a << a << a << " " << a << a << a << endl;

# cout << " " << a << a << a << " " << a << a << a << " " << a << a << a << " " << a << a << a << " " << a << a << a << endl;

# cout << " " << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << " " << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << " " << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << " " << a << a << a << endl;

# cout << " " << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << " " << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << " " << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << " " << a << a << a << endl;

# cout << " " << a << a << a << " " << a << a << a << " " << a << a << a << " " << a << a << a << " " << a << a << a << endl;

# cout << " " << a << a << a << " " << a << a << a << " " << a << a << a << " " << a << a << a << " " << a << a << a << endl;

# cout << " " << a << a << a << " " << a << a << a << " " << a << a << a << " " << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << " " << a << a << a << endl;

# cout << " " << a << a << a << " " << a << a << a << " " << a << a << a << " " << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << " " << a << a << a << endl;

# cout << " " << endl;

# cout << " " << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << " " << a << a << a << a << a << a << a << a << a << a << a << a << a << a << " " << a << a << a << a << a << a << a << a << a << " " << a << a << a << " " << a << a << a << a << a << a << a << a << a << a << a << a << a << " " << a << a << a << a << a << a << a << a << a << a << a << a << a << a << endl;

# cout << " " << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << a << " " << a << a << a << a << a << a << a << a << a << a << a << a << a << a << " " << a << a << a << a << a << a << a << a << a << " " << a << a << a << " " << a << a << a << a << a << a << a << a << a << a << a << a << a << " " << a << a << a << a << a << a << a << a << a << a << a << a << a << a << endl;

# cout << " " << a << a << a << " " << a << a << a << a << " " << a << a << a << " " << a << a << a << " " << a << a << a << " " << a << a << a << " " << a << a << a << " " << a << a << a << " " << a << a << a << " " << a << a << a << " " << a << a << a << endl;

# cout << " " << a << a << a << " " << a << a << a << a << " " << a << a << a << " " << a << a << a << " " << a << a << a << " " << a << a << a << " " << a << a << a << " " << a << a << a << " " << a << a << a << " " << a << a << a << " " << a << a << a << endl;

# cout << " " << a << a << a << " " << a << a << a << a << " " << a << a << a << " " << a << a << a << a << a << a << a << a << a << a << a << a << a << a << " " << a << a << a << " " << a << a << a << " " << a << a << a << " " << a << a << a << " " << a << a << a << a << a << a << a << a << a << a << a << a << a << a << endl;

# cout << " " << a << a << a << " " << a << a << a << a << " " << a << a << a << " " << a << a << a << a << a << a << a << a << a << a << a << a << a << a << " " << a << a << a << " " << a << a << a << " " << a << a << a << " " << a << a << a << " " << a << a << a << a << a << a << a << a << a << a << a << a << a << a << endl;

# cout << " " << a << a << a << " " << a << a << a << a << " " << a << a << a << " " << a << a << a << " " << a << a << a << " " << a << a << a << " " << a << a << a << " " << a << a << a << " " << a << a << a << " " << a << a << a << " " << a << a << a << endl;

# cout << " " << a << a << a << " " << a << a << a << a << " " << a << a << a << " " << a << a << a << " " << a << a << a << " " << a << a << a << " " << a << a << a << " " << a << a << a << " " << a << a << a << " " << a << a << a << " " << a << a << a << endl;

# cout << " " << a << a << a << " " << a << a << a << a << " " << a << a << a << " " << a << a << a << " " << a << a << a << " " << a << a << a << " " << a << a << a << a << a << a << a << a << a << " " << a << a << a << a << a << a << a << a << a << a << a << a << a << " " << a << a << a << " " << a << a << a << endl;

# cout << " " << a << a << a << " " << a << a << a << a << " " << a << a << a << " " << a << a << a << " " << a << a << a << " " << a << a << a << " " << a << a << a << a << a << a << a << a << a << " " << a << a << a << a << a << a << a << a << a << a << a << a << a << " " << a << a << a << " " << a << a << a << endl;

# }

# // top header displays at the top

# void topHeader()

# {

# cout << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << endl;

# cout << box2 << box2 << box2 << " " << box2 << box2 << box2 << endl;

# cout << " " << a << a << a << a << a << a << a << a << " " << a << a << a << a << a << a << a << a << " " << a << a << a << a << a << a << a << a << " " << a << a << a << a << a << a << a << a << " " << a << a << a << a << a << a << a << a << a << a << " " << a << a << a << a << a << a << a << a << " " << a << a << a << a << a << a << " " << a << a << " " << a << a << a << a << a << a << " " << a << a << a << a << a << a << a << a << endl;

# cout << " " << a << a << " "

# << " " << a << a << " " << a << a << " " << a << a << " "

# << " "

# << " " << a << a << " "

# << " " << a << a << " " << a << a << " " << a << a << " " << a << a << " " << a << a << " " << a << a << " " << a << a << " " << a << a << " "

# << " " << a << a << " "

# << " " << a << a << " " << a << a << endl;

# cout << " " << a << a << a << a << a << a << a << a << " " << a << a << a << a << a << a << a << a << " " << a << a << a << a << a << a << a << a << " "

# << " " << a << a << " "

# << " " << a << a << " " << a << a << " " << a << a << " " << a << a << a << a << a << a << a << a << " " << a << a << " " << a << a << " " << a << a << " "

# << " " << a << a << " "

# << " " << a << a << a << a << a << a << a << a << endl;

# cout << " " << a << a << " "

# << " " << a << a << " " << a << a << " "

# << " " << a << a << " "

# << " " << a << a << " "

# << " " << a << a << " " << a << a << " " << a << a << " " << a << a << " " << a << a << " " << a << a << " " << a << a << " " << a << a << " "

# << " " << a << a << " "

# << " " << a << a << " " << a << a << endl;

# cout << " " << a << a << " "

# << " " << a << a << " " << a << a << " " << a << a << a << a << a << a << a << a << " "

# << " " << a << a << " "

# << " " << a << a << " " << a << a << " " << a << a << " " << a << a << " " << a << a << " " << a << a << " " << a << a << a << a << a << a << " " << a << a << a << a << a << a << " " << a << a << " " << a << a << endl;

# cout << box2 << box2 << box2 << " " << box2 << box2 << box2 << endl;

# cout << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << endl;

# }

# // print the status of the game

# void printStatus()

# {

# SetConsoleTextAttribute(hConsole, 10);

# gotoxy(109, 38);

# cout << status;

# }

# // print the health of the main Character

# void printHealth()

# {

# SetConsoleTextAttribute(hConsole, 10);

# gotoxy(75, 38);

# cout << healthCounter;

# }

# // print the Name of the Game

# void printGameName()

# {

# SetConsoleTextAttribute(hConsole, 2);

# gotoxy(18, 38);

# cout << gameName;

# }

# // Generates the lift # 1

# void generateLift1()

# {

# SetConsoleTextAttribute(hConsole, 6);

# int x = 36;

# int y = 7;

# for (int i = 0; i < 7; i++)

# {

# gotoxy(x, y);

# cout << "|";

# for (int i = x + 1; i <= x + 3; i++)

# {

# gotoxy(i, y);

# cout << lift;

# }

# gotoxy(x + 4, y);

# cout << "|";

# y++;

# }

# }

# // Generates the lift # 2

# void generateLift2()

# {

# SetConsoleTextAttribute(hConsole, 6);

# int x = 81;

# int y = 7;

# for (int i = 0; i < 7; i++)

# {

# gotoxy(x, y);

# cout << "|";

# for (int i = x + 1; i <= x + 3; i++)

# {

# gotoxy(i, y);

# cout << lift;

# }

# gotoxy(x + 4, y);

# cout << "|";

# y++;

# }

# }

# // Generates the lift # 3

# void generateLift3()

# {

# SetConsoleTextAttribute(hConsole, 6);

# int x = 51;

# int y = 14;

# for (int i = 0; i < 7; i++)

# {

# gotoxy(x, y);

# cout << "|";

# for (int i = x + 1; i <= x + 3; i++)

# {

# gotoxy(i, y);

# cout << lift;

# }

# gotoxy(x + 4, y);

# cout << "|";

# y++;

# }

# }

# // Generates the lift # 4

# void generateLift4()

# {

# SetConsoleTextAttribute(hConsole, 6);

# int x = 67;

# int y = 14;

# for (int i = 0; i < 7; i++)

# {

# gotoxy(x, y);

# cout << "|";

# for (int i = x + 1; i <= x + 3; i++)

# {

# gotoxy(i, y);

# cout << lift;

# }

# gotoxy(x + 4, y);

# cout << "|";

# y++;

# }

# }

# // Generates the lift # 5

# void generateLift5()

# {

# SetConsoleTextAttribute(hConsole, 6);

# int x = 33;

# int y = 21;

# for (int i = 0; i < 7; i++)

# {

# gotoxy(x, y);

# cout << "|";

# for (int i = x + 1; i <= x + 3; i++)

# {

# gotoxy(i, y);

# cout << lift;

# }

# gotoxy(x + 4, y);

# cout << "|";

# y++;

# }

# }

# // Generates the lift # 6

# void generateLift6()

# {

# SetConsoleTextAttribute(hConsole, 6);

# int x = 84;

# int y = 21;

# for (int i = 0; i < 7; i++)

# {

# gotoxy(x, y);

# cout << "|";

# for (int i = x + 1; i <= x + 3; i++)

# {

# gotoxy(i, y);

# cout << lift;

# }

# gotoxy(x + 4, y);

# cout << "|";

# y++;

# }

# }

# // Generates the lift # 7

# void generateLift7()

# {

# SetConsoleTextAttribute(hConsole, 6);

# int x = 59;

# int y = 28;

# for (int i = 0; i < 7; i++)

# {

# gotoxy(x, y);

# cout << "|";

# for (int i = x + 1; i <= x + 3; i++)

# {

# gotoxy(i, y);

# cout << lift;

# }

# gotoxy(x + 4, y);

# cout << "|";

# y++;

# }

# }

# void printMaze()

# {

# // 87

# SetConsoleTextAttribute(hConsole, 3);

# cout << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << endl;

# cout << box3 << " " << box3 << endl;

# cout << box3 << " " << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << " " << box3 << endl;

# cout << box3 << " " << box2 << " " << box2 << " " << box3 << endl;

# cout << box3 << " " << box2 << " " << box2 << " " << box3 << endl;

# cout << box3 << " " << box2 << " " << box2 << " " << box3 << endl;

# cout << box3 << " " << box2 << " " << box2 << " " << box3 << endl;

# cout << box3 << " " << box2 << " " << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << " " << box2 << " " << box3 << endl;

# cout << box3 << " " << box2 << " " << box2 << " " << box3 << endl;

# cout << box3 << " " << box2 << " " << box2 << " " << box3 << endl;

# cout << box3 << " " << box2 << " " << box2 << " " << box3 << endl;

# cout << box3 << " " << box2 << " " << box2 << " " << box3 << endl;

# cout << box3 << " " << box2 << " " << box2 << " " << box3 << endl;

# cout << box3 << " " << box2 << " " << box2 << " " << box3 << endl;

# cout << box3 << " " << box2 << " " << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << " " << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << " " << box2 << " " << box3 << endl;

# cout << box3 << " " << box2 << " " << box2 << " " << box3 << endl;

# cout << box3 << " " << box2 << " " << box2 << " " << box3 << endl;

# cout << box3 << " " << box2 << " " << box2 << " " << box3 << endl;

# cout << box3 << " " << box2 << " " << box2 << " " << box3 << endl;

# cout << box3 << " " << box2 << " " << box2 << " " << box3 << endl;

# cout << box3 << " " << box2 << " " << box2 << " " << box3 << endl;

# cout << box3 << " " << box2 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << " " << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << " " << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << " " << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box2 << " " << box3 << endl;

# cout << box3 << " " << box2 << " " << box2 << " " << box3 << endl;

# cout << box3 << " " << box2 << " " << box2 << " " << box3 << endl;

# cout << box3 << " " << box2 << " " << box2 << " " << box3 << endl;

# cout << box3 << " " << box2 << " " << box2 << " " << box3 << endl;

# cout << box3 << " " << box2 << " " << box2 << " " << box3 << endl;

# cout << box3 << " " << box2 << " " << box2 << " " << box3 << endl;

# cout << box3 << " " << box2 << " " << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << " " << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << " " << box2 << " " << box3 << endl;

# cout << box3 << " " << box2 << " " << box2 << " " << box3 << endl;

# cout << box3 << " " << box2 << " " << box2 << " " << box3 << endl;

# cout << box3 << " " << box2 << " " << box2 << " " << box3 << endl;

# cout << box3 << " " << box2 << " " << box2 << " " << box3 << endl;

# cout << box3 << " " << box2 << " " << box2 << " " << box3 << endl;

# cout << box3 << " " << box2 << " " << box2 << " " << box3 << endl;

# cout << box3 << " " << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << " " << box3 << endl;

# cout << box3 << " " << box3 << endl;

# cout << box3 << " " << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << box1 << " " << box3 << endl;

# cout << box3 << " " << box1 << " " << box1 << " Health : STATUS : " << box3 << endl;

# cout << box3 << " " << box1 << " " << box1 << " " << box3 << endl;

# cout << box3 << " " << box2 << box1 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box2 << box1 << box2 << " " << box3 << endl;

# cout << box3 << " " << box1 << " " << box1 << " " << box3 << endl;

# cout << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << box3 << endl;

# }

# // move the bullet of enemy 1 to left

# void moveBulletLeftEnemy1()

# {

# for (int x = 0; x < bulletCountLeftEnemy1; x++)

# {

# char next = getcharatxy(enemy1BulletLeftX[x] - 1, enemy1BulletLeftY[x]);

# if (next != ' ')

# {

# eraseBulletLeftEnemy1(enemy1BulletLeftX[x], enemy1BulletLeftY[x]);

# removeBulletfromArrayLeftEnemy1(x);

# }

# else

# {

# eraseBulletLeftEnemy1(enemy1BulletLeftX[x], enemy1BulletLeftY[x]);

# enemy1BulletLeftX[x] = enemy1BulletLeftX[x] - 1;

# printBulletLeftEnemy1(enemy1BulletLeftX[x], enemy1BulletLeftY[x]);

# }

# }

# }

# void eraseBulletLeftEnemy1(int x, int y)

# {

# gotoxy(x, y);

# cout << " ";

# }

# void printBulletLeftEnemy1(int x, int y)

# {

# SetConsoleTextAttribute(hConsole, bulletColour);

# gotoxy(x, y);

# cout << enemyBulletName;

# }

# void removeBulletfromArrayLeftEnemy1(int index)

# {

# for (int x = index; x < bulletCountLeftEnemy1 - 1; x++)

# {

# enemy1BulletLeftX[x] = enemy1BulletLeftX[x + 1];

# enemy1BulletLeftY[x] = enemy1BulletLeftY[x + 1];

# }

# bulletCountLeftEnemy1--;

# }

# // move the bullet of enemy 2 to left

# void moveBulletLeftEnemy2()

# {

# for (int x = 0; x < bulletCountLeftEnemy2; x++)

# {

# char next = getcharatxy(enemy2BulletLeftX[x] - 1, enemy2BulletLeftY[x]);

# if (next != ' ')

# {

# eraseBulletLeftEnemy2(enemy2BulletLeftX[x], enemy2BulletLeftY[x]);

# removeBulletfromArrayLeftEnemy2(x);

# }

# else

# {

# eraseBulletLeftEnemy2(enemy2BulletLeftX[x], enemy2BulletLeftY[x]);

# enemy2BulletLeftX[x] = enemy2BulletLeftX[x] - 1;

# printBulletLeftEnemy2(enemy2BulletLeftX[x], enemy2BulletLeftY[x]);

# }

# }

# }

# void eraseBulletLeftEnemy2(int x, int y)

# {

# gotoxy(x, y);

# cout << " ";

# }

# void printBulletLeftEnemy2(int x, int y)

# {

# SetConsoleTextAttribute(hConsole, bulletColour);

# gotoxy(x, y);

# cout << enemyBulletName;

# }

# void removeBulletfromArrayLeftEnemy2(int index)

# {

# for (int x = index; x < bulletCountLeftEnemy2 - 1; x++)

# {

# enemy2BulletLeftX[x] = enemy2BulletLeftX[x + 1];

# enemy2BulletLeftY[x] = enemy2BulletLeftY[x + 1];

# }

# bulletCountLeftEnemy2--;

# }

# // move the bullet of enemy 3 to left

# void moveBulletLeftEnemy3()

# {

# for (int x = 0; x < bulletCountLeftEnemy3; x++)

# {

# char next = getcharatxy(enemy3BulletLeftX[x] - 1, enemy3BulletLeftY[x]);

# if (next != ' ')

# {

# eraseBulletLeftEnemy3(enemy3BulletLeftX[x], enemy3BulletLeftY[x]);

# removeBulletfromArrayLeftEnemy3(x);

# }

# else

# {

# eraseBulletLeftEnemy3(enemy3BulletLeftX[x], enemy3BulletLeftY[x]);

# enemy3BulletLeftX[x] = enemy3BulletLeftX[x] - 1;

# printBulletLeftEnemy3(enemy3BulletLeftX[x], enemy3BulletLeftY[x]);

# }

# }

# }

# void eraseBulletLeftEnemy3(int x, int y)

# {

# gotoxy(x, y);

# cout << " ";

# }

# void printBulletLeftEnemy3(int x, int y)

# {

# SetConsoleTextAttribute(hConsole, bulletColour);

# gotoxy(x, y);

# cout << enemyBulletName;

# }

# void removeBulletfromArrayLeftEnemy3(int index)

# {

# for (int x = index; x < bulletCountLeftEnemy3 - 1; x++)

# {

# enemy3BulletLeftX[x] = enemy3BulletLeftX[x + 1];

# enemy3BulletLeftY[x] = enemy3BulletLeftY[x + 1];

# }

# bulletCountLeftEnemy3--;

# }

# // move the bullet of enemy 4 to left

# void moveBulletLeftEnemy4()

# {

# for (int x = 0; x < bulletCountLeftEnemy4; x++)

# {

# char next = getcharatxy(enemy4BulletLeftX[x] - 1, enemy4BulletLeftY[x]);

# if (next != ' ')

# {

# eraseBulletLeftEnemy4(enemy4BulletLeftX[x], enemy4BulletLeftY[x]);

# removeBulletfromArrayLeftEnemy4(x);

# }

# else

# {

# eraseBulletLeftEnemy4(enemy4BulletLeftX[x], enemy4BulletLeftY[x]);

# enemy4BulletLeftX[x] = enemy4BulletLeftX[x] - 1;

# printBulletLeftEnemy4(enemy4BulletLeftX[x], enemy4BulletLeftY[x]);

# }

# }

# }

# void eraseBulletLeftEnemy4(int x, int y)

# {

# gotoxy(x, y);

# cout << " ";

# }

# void printBulletLeftEnemy4(int x, int y)

# {

# SetConsoleTextAttribute(hConsole, bulletColour);

# gotoxy(x, y);

# cout << enemyBulletName;

# }

# void removeBulletfromArrayLeftEnemy4(int index)

# {

# for (int x = index; x < bulletCountLeftEnemy4 - 1; x++)

# {

# enemy4BulletLeftX[x] = enemy4BulletLeftX[x + 1];

# enemy4BulletLeftY[x] = enemy4BulletLeftY[x + 1];

# }

# bulletCountLeftEnemy4--;

# }

# // move the bullet of enemy1 to right

# void moveBulletRightEnemy1()

# {

# for (int x = 0; x < bulletCountRightEnemy1; x++)

# {

# char next = getcharatxy(enemy1BulletRightX[x] + 1, enemy1BulletRightY[x]);

# if (next != ' ')

# {

# eraseBulletRightEnemy1(enemy1BulletRightX[x], enemy1BulletRightY[x]);

# removeBulletfromArrayRightEnemy1(x);

# }

# else

# {

# eraseBulletRightEnemy1(enemy1BulletRightX[x], enemy1BulletRightY[x]);

# enemy1BulletRightX[x] = enemy1BulletRightX[x] + 1;

# printBulletRightEnemy1(enemy1BulletRightX[x], enemy1BulletRightY[x]);

# }

# }

# }

# void printBulletRightEnemy1(int x, int y)

# {

# SetConsoleTextAttribute(hConsole, bulletColour);

# gotoxy(x, y);

# cout << enemyBulletName;

# }

# void eraseBulletRightEnemy1(int x, int y)

# {

# gotoxy(x, y);

# cout << " ";

# }

# void removeBulletfromArrayRightEnemy1(int index)

# {

# for (int x = index; x < bulletCountRightEnemy1 - 1; x++)

# {

# enemy1BulletRightX[x] = enemy1BulletRightX[x + 1];

# enemy1BulletRightY[x] = enemy1BulletRightY[x + 1];

# }

# bulletCountRightEnemy1--;

# }

# // move the bullet of enemy2 to Right

# void moveBulletRightEnemy2()

# {

# for (int x = 0; x < bulletCountRightEnemy2; x++)

# {

# char next = getcharatxy(enemy2BulletRightX[x] + 1, enemy2BulletRightY[x]);

# if (next != ' ')

# {

# eraseBulletRightEnemy2(enemy2BulletRightX[x], enemy2BulletRightY[x]);

# removeBulletfromArrayRightEnemy2(x);

# }

# else

# {

# eraseBulletRightEnemy2(enemy2BulletRightX[x], enemy2BulletRightY[x]);

# enemy2BulletRightX[x] = enemy2BulletRightX[x] + 1;

# printBulletRightEnemy2(enemy2BulletRightX[x], enemy2BulletRightY[x]);

# }

# }

# }

# void printBulletRightEnemy2(int x, int y)

# {

# SetConsoleTextAttribute(hConsole, bulletColour);

# gotoxy(x, y);

# cout << enemyBulletName;

# }

# void eraseBulletRightEnemy2(int x, int y)

# {

# gotoxy(x, y);

# cout << " ";

# }

# void removeBulletfromArrayRightEnemy2(int index)

# {

# for (int x = index; x < bulletCountRightEnemy2 - 1; x++)

# {

# enemy2BulletRightX[x] = enemy2BulletRightX[x + 1];

# enemy2BulletRightY[x] = enemy2BulletRightY[x + 1];

# }

# bulletCountRightEnemy2--;

# }

# // move the bullet of enemy3 to Right

# void moveBulletRightEnemy3()

# {

# for (int x = 0; x < bulletCountRightEnemy3; x++)

# {

# char next = getcharatxy(enemy3BulletRightX[x] + 1, enemy3BulletRightY[x]);

# if (next != ' ')

# {

# eraseBulletRightEnemy3(enemy3BulletRightX[x], enemy3BulletRightY[x]);

# removeBulletfromArrayRightEnemy3(x);

# }

# else

# {

# eraseBulletRightEnemy3(enemy3BulletRightX[x], enemy3BulletRightY[x]);

# enemy3BulletRightX[x] = enemy3BulletRightX[x] + 1;

# printBulletRightEnemy3(enemy3BulletRightX[x], enemy3BulletRightY[x]);

# }

# }

# }

# void printBulletRightEnemy3(int x, int y)

# {

# SetConsoleTextAttribute(hConsole, bulletColour);

# gotoxy(x, y);

# cout << enemyBulletName;

# }

# void eraseBulletRightEnemy3(int x, int y)

# {

# gotoxy(x, y);

# cout << " ";

# }

# void removeBulletfromArrayRightEnemy3(int index)

# {

# for (int x = index; x < bulletCountRightEnemy3 - 1; x++)

# {

# enemy3BulletRightX[x] = enemy3BulletRightX[x + 1];

# enemy3BulletRightY[x] = enemy3BulletRightY[x + 1];

# }

# bulletCountRightEnemy3--;

# }

# // move the bullet of enemy4 to Right

# void moveBulletRightEnemy4()

# {

# for (int x = 0; x < bulletCountRightEnemy4; x++)

# {

# char next = getcharatxy(enemy4BulletRightX[x] + 1, enemy4BulletRightY[x]);

# if (next != ' ')

# {

# eraseBulletRightEnemy4(enemy4BulletRightX[x], enemy4BulletRightY[x]);

# removeBulletfromArrayRightEnemy4(x);

# }

# else

# {

# eraseBulletRightEnemy4(enemy4BulletRightX[x], enemy4BulletRightY[x]);

# enemy4BulletRightX[x] = enemy4BulletRightX[x] + 1;

# printBulletRightEnemy4(enemy4BulletRightX[x], enemy4BulletRightY[x]);

# }

# }

# }

# void printBulletRightEnemy4(int x, int y)

# {

# SetConsoleTextAttribute(hConsole, bulletColour);

# gotoxy(x, y);

# cout << enemyBulletName;

# }

# void eraseBulletRightEnemy4(int x, int y)

# {

# gotoxy(x, y);

# cout << " ";

# }

# void removeBulletfromArrayRightEnemy4(int index)

# {

# for (int x = index; x < bulletCountRightEnemy4 - 1; x++)

# {

# enemy4BulletRightX[x] = enemy4BulletRightX[x + 1];

# enemy4BulletRightY[x] = enemy4BulletRightY[x + 1];

# }

# bulletCountRightEnemy4--;

# }

# //-----------------------------------------------

# // Enemy1 Generate Bullet Left

# void generateBulletEnemy1Left()

# {

# SetConsoleTextAttribute(hConsole, bulletColour);

# enemy1BulletLeftX[bulletCountLeftEnemy1] = e1nemyX - 1;

# enemy1BulletLeftY[bulletCountLeftEnemy1] = e1nemyY;

# char next = getcharatxy(e1nemyX - 1, e1nemyY);

# if (next == ' ')

# {

# gotoxy(e1nemyX - 1, e1nemyY);

# cout << enemyBulletName;

# bulletCountLeftEnemy1++;

# }

# }

# // Enemy2 Generate Bullet Left

# void generateBulletEnemy2Left()

# {

# SetConsoleTextAttribute(hConsole, bulletColour);

# enemy2BulletLeftX[bulletCountLeftEnemy2] = e2nemyX - 1;

# enemy2BulletLeftY[bulletCountLeftEnemy2] = e2nemyY;

# char next = getcharatxy(e2nemyX - 1, e2nemyY);

# if (next == ' ')

# {

# gotoxy(e2nemyX - 1, e2nemyY);

# cout << enemyBulletName;

# bulletCountLeftEnemy2++;

# }

# }

# // Enemy3 Generate Bullet Left

# void generateBulletEnemy3Left()

# {

# SetConsoleTextAttribute(hConsole, bulletColour);

# enemy3BulletLeftX[bulletCountLeftEnemy3] = e3nemyX - 1;

# enemy3BulletLeftY[bulletCountLeftEnemy3] = e3nemyY;

# char next = getcharatxy(e3nemyX - 1, e3nemyY);

# if (next == ' ')

# {

# gotoxy(e3nemyX - 1, e3nemyY);

# cout << enemyBulletName;

# bulletCountLeftEnemy3++;

# }

# }

# // Enemy4 Generate Bullet Left

# void generateBulletEnemy4Left()

# {

# SetConsoleTextAttribute(hConsole, bulletColour);

# enemy4BulletLeftX[bulletCountLeftEnemy4] = e4nemyX - 1;

# enemy4BulletLeftY[bulletCountLeftEnemy4] = e4nemyY;

# char next = getcharatxy(e4nemyX - 1, e4nemyY);

# if (next == ' ')

# {

# gotoxy(e4nemyX - 1, e4nemyY);

# cout << enemyBulletName;

# bulletCountLeftEnemy4++;

# }

# }

# // Enemy1 Generate Bullet Right

# void generateBulletEnemy1Right()

# {

# SetConsoleTextAttribute(hConsole, bulletColour);

# enemy1BulletRightX[bulletCountRightEnemy1] = e1nemyX + 6;

# enemy1BulletRightY[bulletCountRightEnemy1] = e1nemyY;

# char next = getcharatxy(e1nemyX + 6, e1nemyY);

# if (next == ' ')

# {

# gotoxy(e1nemyX + 6, e1nemyY);

# cout << enemyBulletName;

# bulletCountRightEnemy1++;

# }

# }

# // Enemy2 Generate Bullet Right

# void generateBulletEnemy2Right()

# {

# SetConsoleTextAttribute(hConsole, bulletColour);

# enemy2BulletRightX[bulletCountRightEnemy2] = e2nemyX + 6;

# enemy2BulletRightY[bulletCountRightEnemy2] = e2nemyY;

# char next = getcharatxy(e2nemyX + 6, e2nemyY);

# if (next == ' ')

# {

# gotoxy(e2nemyX + 6, e2nemyY);

# cout << enemyBulletName;

# bulletCountRightEnemy2++;

# }

# }

# // Enemy3 Generate Bullet Right

# void generateBulletEnemy3Right()

# {

# SetConsoleTextAttribute(hConsole, bulletColour);

# enemy3BulletRightX[bulletCountRightEnemy3] = e3nemyX + 6;

# enemy3BulletRightY[bulletCountRightEnemy3] = e3nemyY;

# char next = getcharatxy(e3nemyX + 6, e3nemyY);

# if (next == ' ')

# {

# gotoxy(e3nemyX + 6, e3nemyY);

# cout << enemyBulletName;

# bulletCountRightEnemy3++;

# }

# }

# // Enemy3 Generate Bullet Right

# void generateBulletEnemy4Right()

# {

# SetConsoleTextAttribute(hConsole, bulletColour);

# enemy4BulletRightX[bulletCountRightEnemy4] = e4nemyX + 6;

# enemy4BulletRightY[bulletCountRightEnemy4] = e4nemyY;

# char next = getcharatxy(e4nemyX + 6, e4nemyY);

# if (next == ' ')

# {

# gotoxy(e4nemyX + 6, e4nemyY);

# cout << enemyBulletName;

# bulletCountRightEnemy4++;

# }

# }

# //-----------------------Generates the bullet of the main character to RIGHTTTTT

# void generateBulletRight()

# {

# SetConsoleTextAttribute(hConsole, bulletColour);

# char bulletRight = 16;

# bulletRightX[bulletCountRight] = pX + 5;

# bulletRightY[bulletCountRight] = pY + 1;

# char next = getcharatxy(pX + 5, pY + 1);

# if (next == ' ')

# {

# gotoxy(pX + 5, pY + 1);

# cout << bulletRight;

# bulletCountRight++;

# }

# }

# void moveBulletRight()

# {

# for (int x = 0; x < bulletCountRight; x++)

# {

# char next = getcharatxy(bulletRightX[x] + 1, bulletRightY[x]);

# if (next != ' ')

# {

# eraseBulletRight(bulletRightX[x], bulletRightY[x]);

# removeBulletfromArrayRight(x);

# }

# else

# {

# eraseBulletRight(bulletRightX[x], bulletRightY[x]);

# bulletRightX[x] = bulletRightX[x] + 1;

# printBulletRight(bulletRightX[x], bulletRightY[x]);

# }

# }

# }

# void eraseBulletRight(int x, int y)

# {

# gotoxy(x, y);

# cout << " ";

# }

# void printBulletRight(int x, int y)

# {

# SetConsoleTextAttribute(hConsole, bulletColour);

# char bulletRight = 16;

# gotoxy(x, y);

# cout << bulletRight;

# }

# void removeBulletfromArrayRight(int index)

# {

# for (int x = index; x < bulletCountRight - 1; x++)

# {

# bulletRightX[x] = bulletRightX[x + 1];

# bulletRightY[x] = bulletRightY[x + 1];

# }

# bulletCountRight--;

# }

# //------------------Collision woth the enemy

# void bulletCollisionWithEnemyRight()

# {

# for (int x = 0; x < bulletCountRight; x++)

# {

# if (bulletRightX[x] + 1 == e1nemyX && (bulletRightY[x] == e1nemyY || bulletRightY[x] == e1nemyY + 1))

# {

# addScore();

# eraseBulletRight(bulletRightX[x], bulletRightY[x]);

# removeBulletfromArrayRight(x);

# enemy1Health--;

# }

# if (bulletRightX[x] + 1 == e2nemyX && (bulletRightY[x] == e2nemyY || bulletRightY[x] == e2nemyY + 1))

# {

# addScore();

# eraseBulletRight(bulletRightX[x], bulletRightY[x]);

# removeBulletfromArrayRight(x);

# enemy2Health--;

# }

# if (bulletRightX[x] + 1 == e3nemyX && (bulletRightY[x] == e3nemyY || bulletRightY[x] == e3nemyY + 1))

# {

# addScore();

# eraseBulletRight(bulletRightX[x], bulletRightY[x]);

# removeBulletfromArrayRight(x);

# enemy3Health--;

# }

# if (bulletRightX[x] + 1 == e4nemyX && (bulletRightY[x] == e4nemyY || bulletRightY[x] == e4nemyY + 1))

# {

# addScore();

# eraseBulletRight(bulletRightX[x], bulletRightY[x]);

# removeBulletfromArrayRight(x);

# enemy4Health--;

# }

# }

# }

# //---------move bullet Left------------------------------------------------------------------------------------------------------------

# void generateBulletLeft()

# {

# SetConsoleTextAttribute(hConsole, bulletColour);

# char bulletLeft = 17;

# bulletLeftX[bulletCountLeft] = pX - 1;

# bulletLeftY[bulletCountLeft] = pY + 1;

# char next = getcharatxy(pX - 1, pY + 1);

# if (next == ' ')

# {

# gotoxy(pX - 1, pY + 1);

# cout << bulletLeft;

# bulletCountLeft++;

# }

# }

# void moveBulletLeft()

# {

# for (int x = 0; x < bulletCountLeft; x++)

# {

# char next = getcharatxy(bulletLeftX[x] - 1, bulletLeftY[x]);

# if (next != ' ')

# {

# eraseBulletLeft(bulletLeftX[x], bulletLeftY[x]);

# removeBulletfromArrayLeft(x);

# }

# else

# {

# eraseBulletLeft(bulletLeftX[x], bulletLeftY[x]);

# bulletLeftX[x] = bulletLeftX[x] - 1;

# printBulletLeft(bulletLeftX[x], bulletLeftY[x]);

# }

# }

# }

# void eraseBulletLeft(int x, int y)

# {

# gotoxy(x, y);

# cout << " ";

# }

# void printBulletLeft(int x, int y)

# {

# SetConsoleTextAttribute(hConsole, bulletColour);

# char bulletLeft = 17;

# gotoxy(x, y);

# cout << bulletLeft;

# }

# void removeBulletfromArrayLeft(int index)

# {

# for (int x = index; x < bulletCountLeft - 1; x++)

# {

# bulletLeftX[x] = bulletLeftX[x + 1];

# bulletLeftY[x] = bulletLeftY[x + 1];

# }

# bulletCountLeft--;

# }

# //------------------Collision with the enemy Left---------------------------------

# void bulletCollisionWithEnemyLeft()

# {

# for (int x = 0; x < bulletCountLeft; x++)

# {

# if (bulletLeftX[x] - 1 == e1nemyX + 5 && (bulletLeftY[x] == e1nemyY || bulletLeftY[x] == e1nemyY + 1))

# {

# addScore();

# eraseBulletLeft(bulletLeftX[x], bulletLeftY[x]);

# removeBulletfromArrayLeft(x);

# enemy1Health--;

# }

# if (bulletLeftX[x] - 1 == e2nemyX + 5 && (bulletLeftY[x] == e2nemyY || bulletLeftY[x] == e2nemyY + 1))

# {

# addScore();

# eraseBulletLeft(bulletLeftX[x], bulletLeftY[x]);

# removeBulletfromArrayLeft(x);

# enemy2Health--;

# }

# if (bulletLeftX[x] - 1 == e3nemyX + 5 && (bulletLeftY[x] == e3nemyY || bulletLeftY[x] == e3nemyY + 1))

# {

# addScore();

# eraseBulletLeft(bulletLeftX[x], bulletLeftY[x]);

# removeBulletfromArrayLeft(x);

# enemy3Health--;

# }

# if (bulletLeftX[x] - 1 == e4nemyX + 5 && (bulletLeftY[x] == e4nemyY || bulletLeftY[x] == e4nemyY + 1))

# {

# addScore();

# eraseBulletLeft(bulletLeftX[x], bulletLeftY[x]);

# removeBulletfromArrayLeft(x);

# enemy4Health--;

# }

# }

# }

# void removeHealth()

# {

# healthCounter--;

# }

# void addScore()

# {

# score++;

# }

# //-----------moveEnemy1

# void moveEnemy1()

# {

# if (enemyDirection1 == "Right")

# {

# char next = getcharatxy(e1nemyX + 6, e1nemyY);

# char next1 = getcharatxy(e1nemyX + 6, e1nemyY + 2);

# if ((next == ' ') && ((next1 == box1) || (next1 == lift) || (next1 == '|')))

# {

# eraseEnemy1();

# e1nemyX++;

# printEnemy1(e1nemyX);

# }

# if (next1 == ' ')

# {

# enemyDirection1 = "Left";

# }

# generateBulletEnemy1Right();

# }

# if (enemyDirection1 == "Left")

# {

# char next = getcharatxy(e1nemyX - 1, e1nemyY);

# char next1 = getcharatxy(e1nemyX - 1, e1nemyY + 2);

# if ((next == ' ') && ((next1 == box1) || (next1 == lift) || (next1 == '|')))

# {

# eraseEnemy1();

# e1nemyX--;

# printEnemy1(e1nemyX);

# }

# if (next1 == ' ')

# {

# enemyDirection1 = "Right";

# }

# generateBulletEnemy1Left();

# }

# }

# //-----------erase enemy1

# void eraseEnemy1()

# {

# gotoxy(e1nemyX, e1nemyY);

# for (int index = 0; index < 6; index++)

# {

# cout << " ";

# }

# gotoxy(e1nemyX, e1nemyY + 1);

# for (int index = 0; index < 6; index++)

# {

# cout << " ";

# }

# }

# //--------------------Print Enemy 1

# void printEnemy1(int e1nemyX)

# {

# SetConsoleTextAttribute(hConsole, 4);

# gotoxy(e1nemyX, e1nemyY);

# for (int index = 0; index < 6; index++)

# {

# cout << e1nemy1[index];

# }

# gotoxy(e1nemyX, e1nemyY + 1);

# for (int index = 0; index < 6; index++)

# {

# cout << e1nemy2[index];

# }

# }

# //-----------moveEnemy2

# void moveEnemy2()

# {

# if (enemyDirection2 == "Right")

# {

# char next = getcharatxy(e2nemyX + 6, e2nemyY);

# char next1 = getcharatxy(e2nemyX + 6, e2nemyY + 2);

# if ((next == ' ') && ((next1 == box1) || (next1 == lift) || (next1 == '|')))

# {

# eraseEnemy2();

# e2nemyX++;

# printEnemy2(e2nemyX);

# }

# if (next != ' ')

# {

# enemyDirection2 = "Left";

# }

# generateBulletEnemy2Right();

# }

# if (enemyDirection2 == "Left")

# {

# char next = getcharatxy(e2nemyX - 1, e2nemyY);

# char next1 = getcharatxy(e2nemyX - 1, e2nemyY + 2);

# if ((next == ' ') && ((next1 == box1) || (next1 == lift) || (next1 == '|')))

# {

# eraseEnemy2();

# e2nemyX--;

# printEnemy2(e2nemyX);

# }

# if (next != ' ')

# {

# enemyDirection2 = "Right";

# }

# generateBulletEnemy2Left();

# }

# }

# //-----------erase enemy2

# void eraseEnemy2()

# {

# gotoxy(e2nemyX, e2nemyY);

# for (int index = 0; index < 6; index++)

# {

# cout << " ";

# }

# gotoxy(e2nemyX, e2nemyY + 1);

# for (int index = 0; index < 6; index++)

# {

# cout << " ";

# }

# }

# //--------------------Print Enemy 2

# void printEnemy2(int e2nemyX)

# {

# SetConsoleTextAttribute(hConsole, 4);

# gotoxy(e2nemyX, e2nemyY);

# for (int index = 0; index < 6; index++)

# {

# cout << e2nemy1[index];

# }

# gotoxy(e2nemyX, e2nemyY + 1);

# for (int index = 0; index < 6; index++)

# {

# cout << e2nemy2[index];

# }

# }

# //-----------moveEnemy3

# void moveEnemy3()

# {

# if (enemyDirection3 == "Right")

# {

# char next = getcharatxy(e3nemyX + 6, e3nemyY);

# char next1 = getcharatxy(e3nemyX + 6, e3nemyY + 2);

# if ((next == ' ') && ((next1 == box1) || (next1 == lift) || (next1 == '|')))

# {

# eraseEnemy3();

# e3nemyX++;

# printEnemy3(e3nemyX);

# }

# if (next != ' ')

# {

# enemyDirection3 = "Left";

# }

# generateBulletEnemy3Right();

# }

# if (enemyDirection3 == "Left")

# {

# char next = getcharatxy(e3nemyX - 1, e3nemyY);

# char next1 = getcharatxy(e3nemyX - 1, e3nemyY + 2);

# if ((next == ' ') && ((next1 == box1) || (next1 == lift) || (next1 == '|')))

# {

# eraseEnemy3();

# e3nemyX--;

# printEnemy3(e3nemyX);

# }

# if (next != ' ')

# {

# enemyDirection3 = "Right";

# }

# generateBulletEnemy3Left();

# }

# }

# //-----------erase enemy3

# void eraseEnemy3()

# {

# gotoxy(e3nemyX, e3nemyY);

# for (int index = 0; index < 6; index++)

# {

# cout << " ";

# }

# gotoxy(e3nemyX, e3nemyY + 1);

# for (int index = 0; index < 6; index++)

# {

# cout << " ";

# }

# }

# //--------------------Print Enemy 3

# void printEnemy3(int e3nemyX)

# {

# SetConsoleTextAttribute(hConsole, 4);

# gotoxy(e3nemyX, e3nemyY);

# for (int index = 0; index < 6; index++)

# {

# cout << e3nemy1[index];

# }

# gotoxy(e3nemyX, e3nemyY + 1);

# for (int index = 0; index < 6; index++)

# {

# cout << e3nemy2[index];

# }

# }

# //-----------moveEnemy4

# void moveEnemy4()

# {

# if (enemyDirection4 == "Right")

# {

# char next = getcharatxy(e4nemyX + 6, e4nemyY);

# char next1 = getcharatxy(e4nemyX + 6, e4nemyY + 2);

# if ((next == ' ') && ((next1 == box1) || (next1 == lift) || (next1 == '|')))

# {

# eraseEnemy4();

# e4nemyX++;

# printEnemy4(e4nemyX);

# }

# if (next != ' ')

# {

# enemyDirection4 = "Left";

# }

# generateBulletEnemy4Right();

# }

# if (enemyDirection4 == "Left")

# {

# char next = getcharatxy(e4nemyX - 1, e4nemyY);

# char next1 = getcharatxy(e4nemyX - 1, e4nemyY + 2);

# if ((next == ' ') && ((next1 == box1) || (next1 == lift) || (next1 == '|')))

# {

# eraseEnemy4();

# e4nemyX--;

# printEnemy4(e4nemyX);

# }

# if (next != ' ')

# {

# enemyDirection4 = "Right";

# }

# generateBulletEnemy4Left();

# }

# }

# //-----------erase enemy 4

# void eraseEnemy4()

# {

# gotoxy(e4nemyX, e4nemyY);

# for (int index = 0; index < 6; index++)

# {

# cout << " ";

# }

# gotoxy(e4nemyX, e4nemyY + 1);

# for (int index = 0; index < 6; index++)

# {

# cout << " ";

# }

# }

# //--------------------Print Enemy 4

# void printEnemy4(int e4nemyX)

# {

# SetConsoleTextAttribute(hConsole, 4);

# gotoxy(e4nemyX, e4nemyY);

# for (int index = 0; index < 6; index++)

# {

# cout << e4nemy1[index];

# }

# gotoxy(e4nemyX, e4nemyY + 1);

# for (int index = 0; index < 6; index++)

# {

# cout << e4nemy2[index];

# }

# }

# //------------------------Move Mario Left

# void moveMarioLeft()

# {

# char next1 = getcharatxy(pX - 1, pY);

# char next2 = getcharatxy(pX - 1, pY + 1);

# char next3 = getcharatxy(pX - 1, pY + 2);

# char next4 = getcharatxy(pX - 1, pY + 3);

# if (((next1 == ' ' && next2 == ' ' && next3 == ' ') || (next1 == '|' && next2 == '|' && next3 == '|') || (next1 == lift && next2 == lift && next3 == lift && next3 == lift)) && (next4 != ' '))

# {

# eraseCharacter();

# pX = pX - 1;

# printCharacter();

# }

# if (next1 == enemyBulletName || next2 == enemyBulletName || next3 == enemyBulletName || next4 == enemyBulletName)

# {

# healthCounter--;

# }

# }

# //------------------------Move Mario Right

# void moveMarioRight()

# {

# char next1 = getcharatxy(pX + 5, pY);

# char next2 = getcharatxy(pX + 5, pY + 1);

# char next3 = getcharatxy(pX + 5, pY + 2);

# char next4 = getcharatxy(pX + 5, pY + 3);

# if (((next1 == ' ' && next2 == ' ' && next3 == ' ') || (next1 == '|' && next2 == '|' && next3 == '|') || (next1 == lift && next2 == lift && next3 == lift && next3 == lift)) && (next4 != ' '))

# {

# eraseCharacter();

# pX = pX + 1;

# printCharacter();

# }

# if (next1 == enemyBulletName || next2 == enemyBulletName || next3 == enemyBulletName || next4 == enemyBulletName)

# {

# healthCounter--;

# }

# }

# //------------------------Move Mario Up

# void moveMarioUp()

# {

# char next1 = getcharatxy(pX, pY - 1);

# char next2 = getcharatxy(pX + 1, pY - 1);

# char next3 = getcharatxy(pX + 2, pY - 1);

# char next4 = getcharatxy(pX + 3, pY - 1);

# char next5 = getcharatxy(pX + 4, pY - 1);

# char next6 = getcharatxy(pX + 5, pY);

# char next7 = getcharatxy(pX - 1, pY);

# char next8 = getcharatxy(pX + 5, pY + 1);

# char next9 = getcharatxy(pX - 1, pY + 1);

# char next10 = getcharatxy(pX + 5, pY + 2);

# char next11 = getcharatxy(pX - 1, pY + 2);

# if ((next1 == '|' && next2 == lift && next3 == lift && next4 == lift && next5 == '|') || (next6 == box1 || next7 == box1) || (next8 == box1 || next9 == box1) || (next10 == box1 || next11 == box1))

# {

# eraseCharacter();

# pY = pY - 1;

# printCharacter();

# }

# }

# //------------------------Move Mario Down

# void moveMarioDown()

# {

# char next1 = getcharatxy(pX, pY + 3);

# char next2 = getcharatxy(pX + 1, pY + 3);

# char next3 = getcharatxy(pX + 2, pY + 3);

# char next4 = getcharatxy(pX + 3, pY + 3);

# char next5 = getcharatxy(pX + 4, pY + 3);

# if ((next1 == '|' && next2 == lift && next3 == lift && next4 == lift && next5 == '|'))

# {

# eraseCharacter();

# pY = pY + 1;

# printCharacter();

# }

# }

# //----------------SCORE Print

# void printScore()

# {

# SetConsoleTextAttribute(hConsole, 10);

# gotoxy(40, 38);

# cout << "Score : " << score;

# }

# void gotoxy(int x, int y)

# {

# COORD coordinate;

# coordinate.X = x;

# coordinate.Y = y;

# SetConsoleCursorPosition(GetStdHandle(STD\_OUTPUT\_HANDLE), coordinate);

# }

# char getcharatxy(short int x, short int y)

# {

# CHAR\_INFO ci;

# COORD xy = {0, 0};

# SMALL\_RECT rect = {x, y, x, y};

# COORD coordbufsize;

# coordbufsize.X = 1;

# coordbufsize.Y = 1;

# return ReadConsoleOutput(GetStdHandle(STD\_OUTPUT\_HANDLE), &ci, coordbufsize, xy, &rect) ? ci.Char.AsciiChar : ' ';

# }

# // -------------------print the main character

# void printCharacter()

# {

# SetConsoleTextAttribute(hConsole, 1);

# gotoxy(pX, pY);

# for (int i = 0; i < 5; i++)

# {

# cout << character1[i];

# }

# gotoxy(pX, pY + 1);

# for (int i = 0; i < 5; i++)

# {

# cout << character2[i];

# }

# gotoxy(pX, pY + 2);

# for (int i = 0; i < 5; i++)

# {

# cout << character3[i];

# }

# }

# //---------------------Erase the main character

# void eraseCharacter()

# {

# gotoxy(pX, pY);

# for (int i = 0; i < 5; i++)

# {

# cout << " ";

# }

# gotoxy(pX, pY + 1);

# for (int i = 0; i < 5; i++)

# {

# cout << " ";

# }

# gotoxy(pX, pY + 2);

# for (int i = 0; i < 5; i++)

# {

# cout << " ";

# }

# }

# void healthDecrement()

# {

# char next = getcharatxy(pX - 1, pY);

# char next1 = getcharatxy(pX + 5, pY);

# char next2 = getcharatxy(pX - 1, pY + 1);

# char next3 = getcharatxy(pX + 5, pY + 1);

# if (next == char(15) || next1 == char(15) || next2 == char(15) || next3 == char(15))

# {

# healthCounter--;

# }

# }

# void storeCoordinates()

# {

# fstream file;

# file.open("CharacterInfo.txt", ios::out);

# file << pX << "\n";

# file << pY << "\n";

# file << e1nemyX << "\n";

# file << e1nemyY << "\n";

# file << e2nemyX << "\n";

# file << e2nemyX << "\n";

# file << e3nemyX << "\n";

# file << e3nemyX << "\n";

# file << e4nemyX << "\n";

# file << e4nemyX << "\n";

# file.close();

# }

# void storeHealth()

# {

# fstream file;

# file.open("Health.txt", ios::out);

# file << healthCounter;

# file.close();

# }

# void storeScore()

# {

# fstream file;

# file.open("StoreScore.txt", ios::out);

# file << score;

# file.close();

# }

# void LoadCoordinate()

# {

# fstream file;

# string px;

# string py;

# string e1X;

# string e2X;

# string e3X;

# string e4X;

# string e1Y;

# string e2Y;

# string e3Y;

# string e4Y;

# file.open("CharacterInfo.txt", ios::in);

# while (getline(file, px) && getline(file, py) && getline(file, e1X) && getline(file, e2X) && getline(file, e3X) && getline(file, e4X) && getline(file, e1Y) && getline(file, e2Y) && getline(file, e3Y) && getline(file, e4Y))

# {

# pX = stoi(px);

# pY = stoi(py);

# e1nemyX = stoi(e1X);

# e2nemyX = stoi(e2X);

# e3nemyX = stoi(e3X);

# e4nemyX = stoi(e4X);

# e1nemyY = stoi(e1Y);

# e2nemyY = stoi(e2Y);

# e3nemyY = stoi(e3Y);

# e4nemyY = stoi(e4Y);

# }

# file.close();

# }

**Student Reg. No. : 2022 R/2021-CS-199 Student Name. : MUHAMMAD FURQAN**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **A-Extensive Evidence** | **B-Convincing Evidence** | **C-Limited Evidence** | **D-No Evidence** | |
| Documenta**C**tio**h**n**ec** Formatting  **Grade:** | **k**A**e**l**d**l th**b**e**y**d**:**ocumentationClick meets all the criteria. | orDtoacpumheernetattoionenteisr texwt.ell formatted but some of the  criteria is not fulfilled. | Documentation is required a lot of improvement. | Documentation is not Available |  |
| **Documentation Formatting Criteria:** In **Binder**, **Title** Page, **Header**-Footers, Font **Style**, Font **Size** all are all consistence and according to given **guidelines**. Project **Poster** is professionally design and well presented | | | | | |
| Documentation Contents **Grade:** | Documentation includes all of the criteria. | Documentation meet more than 80% of the criteria given. | Documentation meet more than 50% of the criteria. | When the documentation meet less than 50% of the criteria. | |
| **Documentation Contents Criteria: Title** Page - **Table** of Contents - Project **Short Description and Story Writing of Game** - **Game Characters** Description - **Rules** & Interactions - **Goal** of the Game **- Screenshot** of the Game - **Data Structures** Used in the Game -  **Functions** Prototype - **Full Code** | | | | | |
| Project  Complexity  **Grade:** | Project has at least 1 Player and 3 enemies. Proper use of gotoxy()  function. Health system, Firing System and lives decreasing system. | Project complexity meet 80% criteria given in extensive evidence | Project complexity meet 50% criteria given in extensive evidence | Project complexity meet less than 50% criteria given in extensive evidence | |
| Randomness  **Grade:** | Objects are produced  randomly in the game. | meet more than 80% of the  criteria given. | meet more than 50% of the  criteria given. | Objects are appearing in the  same pattern | |
| Code Style  **Grade:** | All Code style criteria is followed | All code style criteria followed but some improvements required | lot of improvements required in coding style. | **Did not follow** code style, | |
| **Code Style Criteria:** Consistent code style. Code is well indented. Variable and Function names are well defined.  White Spaces are well used. Comments are added. | | | | | |
| Code  Documentation Mapping **Grade:** | Code and documentation is synchronized. | Code and documentation does not synchronized at **some** places | Code and documentation does not synchronized at **many** places | Code and documentation  **does not** synchronized. | |
| Idea Novelty and Creativity **Grade:** | Idea is unique of the game | Idea is merged by combining other different games | Same idea as a previous game | Could not implement the existing game idea. | |
| Data Structure (2D Arrays) **Grade:** | Data structure is sufficient for the project requirements | Data Structure is sufficient but require improvement to meet project requirements. | Data structure is not sufficient and need a lot of  improvement | Data Structure is not properly identified and declared. | |
| File Handling  **Grade:** | Game maze is loaded and the updated maze is stored in the file | Game maze is loaded and partial data is stored in the file. | Game maze is just loaded but the updated game configuration is not stored in  the maze. | Project do not contain file handling | |
| Modularity  **Grade:** | Meet all Modularity criteria | Meet all Modularity criteria but at some places it is missing | Do not sufficiently meet the modularity criteria. | No modularity or very minimum modularity. | |
| **Modularity criteria:** Functions are defined for each major feature. Functions are independent (identify from parameter list and return types)- There is no global variable defined. Arrays and variables are passed as parameters to the functions. Functions exhibit single responsibility  principle. | | | | | |
| Screen flickering **Grade:** | There is no Screen flickering. | Maze is not flickering but the characters are flickering at great speed | Flickering is done at lot of places | Screen is flickering at all places | |
| Presentation and Demo **Grade:** | Presentation and Demo was 100% working | Presentation and Demo require some  improvements | Presentation and Demo require a lot of improvements | Presentation was not ok and Demo was not working | |
| Student  Understanding with the Code. **Grade:** | Student has complete understanding how the code is working and  knows the concept. | Student has good understand but some place he does not know the  concepts | Student has a very little understand and lack the major concepts. | Student does not have any  level of understanding of the code. | |