Space Shooter

Session: 2022 – 2026

Submitted by:

Shahbaz Ali 2022-CS-27

Supervised by:

Maida Shahid

Department of Computer Science

University of Engineering and Technology

Lahore Pakistan

Short Description and Story Writing of Game

Bahadur lives in a world which was attacked by an alien race called “The Yincas”. Bahadur has a spaceship known as “StarFighter”. He is a highly skilled pilot. He spends his days defending his territory and destroying as many as enemy spaceships he can. The Yincas spaceships also lurk in the space to kill Bahadur.

Despite the danger, Bahadur refuses to give up. He knows that if he kills the Yincas's boss then he will be able to save the world as well as his love ones. So, Bahadur sets out on a quest to destroy particular number of enemy spaceships as given by the headquarters.

As Bahadur navigates the space, he faces many danger and challenges. But he never loses hope and he never gives up. With his bravery and splendid skills, he becomes stronger and more powerful with each victory.

Bahadur eventually succeeds in clearing the target. He becomes a hero, known for his bravery, determination, and never-say-die spirit. The Yincas may try to stop him time and time again, but Bahadur always perseveres and triumphs in the end.

The legend of Bahadur lives on, inspiring new generations of players to defend their world and love ones. Space Shooter game remains a timeless classic that captivates and entertains players of all ages.

Game Characters Description

Player

There is one human player in the game.

Bahadur:

Bahadur is the main character in the game which is seated in a spaceship called “StarFighter”. He is fighter and defend his world and love ones from alien invasion. Bahadur is brave, determined, and has a never-say-die spirit. He is the hero of the game, admired for his bravery and determination in the face of danger.

Enemies

There are 2 types of enemies in the game.

Yincas Spaceship:

Yincas spaceships are the alien spaceships which is determined to kill all human and undertake the world. Though they are weaker than the bosses but they are in a large number.

Alpha Yincas Spaceship:
Alpha Yincas Spaceship is another alien spaceship and it is the boss of this game. After destroying Alpha Yincas Spaceship the game will be cleared.

Game Objects Description

Following are the objects in the game

Laser:

There are installed laser guns in both the ships of Bahadur and Yincas. When the gun is fired the laser can be seen. If it hits the enemy's ship it will be destroyed and if enemy's laser hit player's ship the it will loose a life.

Walls:

Walls are the barriers in the game which Bahadur and the Yincas cannot cross.

Rules & Instructions

Bahadur can destroy the spaceships which are inside the walls. Bahadur loses a life if he collides with the enemy's ship or got hit by laser. Score increases and number of target decreases when Bahadur destroys enemy’s ship.

Goal of the Game

The goal of the game is to fulfill the target which were assigned to Bahadur by the headquarters while avoiding the counter attacks.

Wireframes

Figure 01. Main Screen

Figure 02. Instruction Screen

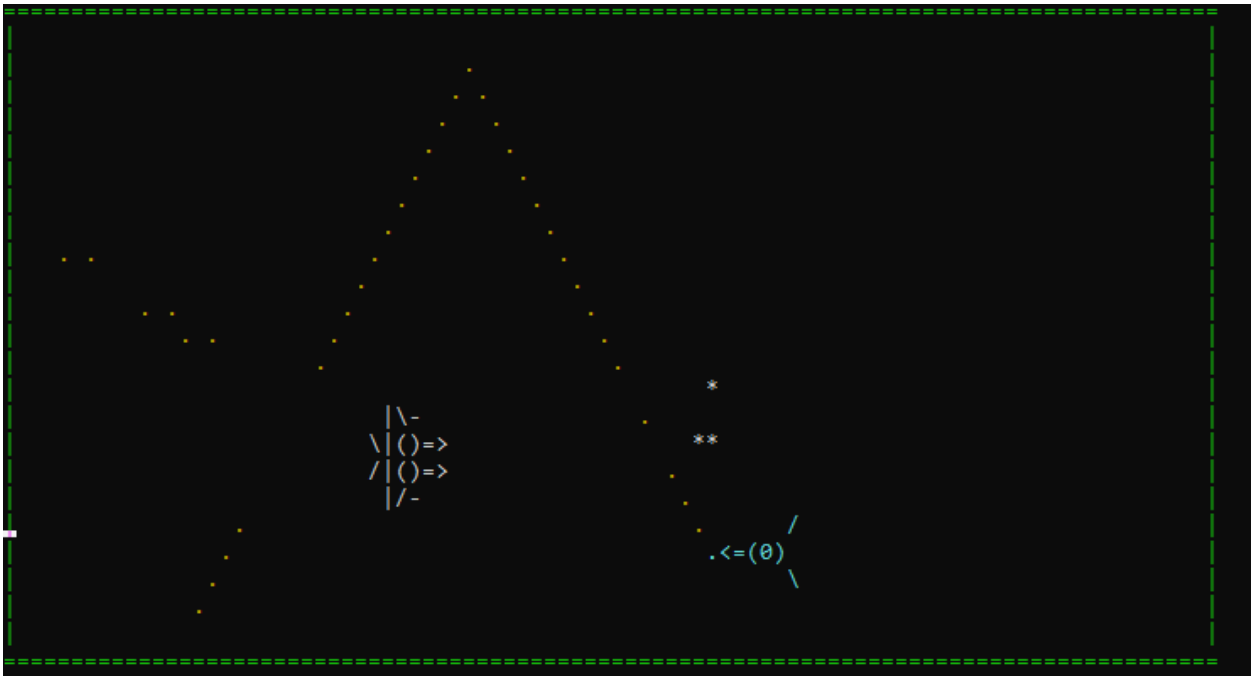


Figure 03. Battle Space Screen

Data Structures (2D Arrays)

```
char heroShip[4][6] = {{' ', '|', '\\', '-', ' ', ' '},
                       {'\\', '|', '(', ')', '=', '>'},
                       {'/', '|', '(', ')', '=', '>'},
                       {' ', '|', '/', '-', ' ', ' '}};

char enemy1[3][6] = {{' ', ' ', ' ', ' ', ' ', '/'},
                     {'<', '=', '(', '0', ')', ' '},
                     {' ', ' ', ' ', ' ', ' ', '\\'}};

char enemy2[3][6] = {{' ', ' ', ' ', ' ', ' ', '/'},
                     {'<', '=', '(', '0', ')', ' '},
                     {' ', ' ', ' ', ' ', ' ', '\\'}};

char enemy3[3][6] = {{' ', ' ', ' ', ' ', ' ', '/'},
                     {'<', '=', '(', '0', ')', ' '},
                     {' ', ' ', ' ', ' ', ' ', '\\'}};
```

```
int bulletX[10000];
int bulletY[10000];
bool isBulletActive[10000];
int enemybulletX[100000];
int enemybulletY[100000];
bool isenemyBulletActive[100000];
```

Function Prototypes

```
void generateColors(); // For Colors
void wall();
void printHeroShip();
void printEnemy1();
void printEnemy2();
void printEnemy3();
void eraseHeroShip();
void moveHeroShipLeft();
void moveHeroShipRight();
void moveHeroShipUp();
void moveHeroShipDown();
void generateBullet();
void generateEnemyBullet();
void generateEnemyBullet2();
void moveBullet();
void moveenemyBullet();
void printBullet(int x, int y);
void printenemyBullet(int x, int y);
void eraseBullet(int x, int y);
void eraseenemyBullet(int x, int y);
void makeBulletInactive(int index);
void makeenemyBulletInactive(int index);
void moveEnemy1();
void moveEnemy2();
void eraseEnemy1();
void eraseEnemy2();
void addScore();
void printScore();
void bulletCollisionWithEnemy();
void bulletCollisionWithEnemy2();
void eraseEnemyAfterCollision();
void startScreen1();
void startScreen2();
void reduceEnemyLife();
```

```
void bulletCollisionWithHero();
void bulletCollisionWithHero2();
void reduceHeroLife();
void reduceEnemyLife2();
void gameover();
bool isWinner();
```

Complete Code

```
#include <iostream>
#include <windows.h>
#include <conio.h>
#include <fstream>
using namespace std;

void generateColors(); // For Colors

void wall();
void printHeroShip();
void printEnemy1();
void printEnemy2();
void printEnemy3();
void eraseHeroShip();
void moveHeroShipLeft();
void moveHeroShipRight();
void moveHeroShipUp();
void moveHeroShipDown();
void generateBullet();
void generateEnemyBullet();
void generateEnemyBullet2();
void moveBullet();
void moveenemyBullet();
void printBullet(int x, int y);
void printenemyBullet(int x, int y);
void eraseBullet(int x, int y);
void eraseenemyBullet(int x, int y);
void makeBulletInactive(int index);
void makeenemyBulletInactive(int index);
void moveEnemy1();
void moveEnemy2();
void eraseEnemy1();
void eraseEnemy2();
void addScore();
void printScore();
```

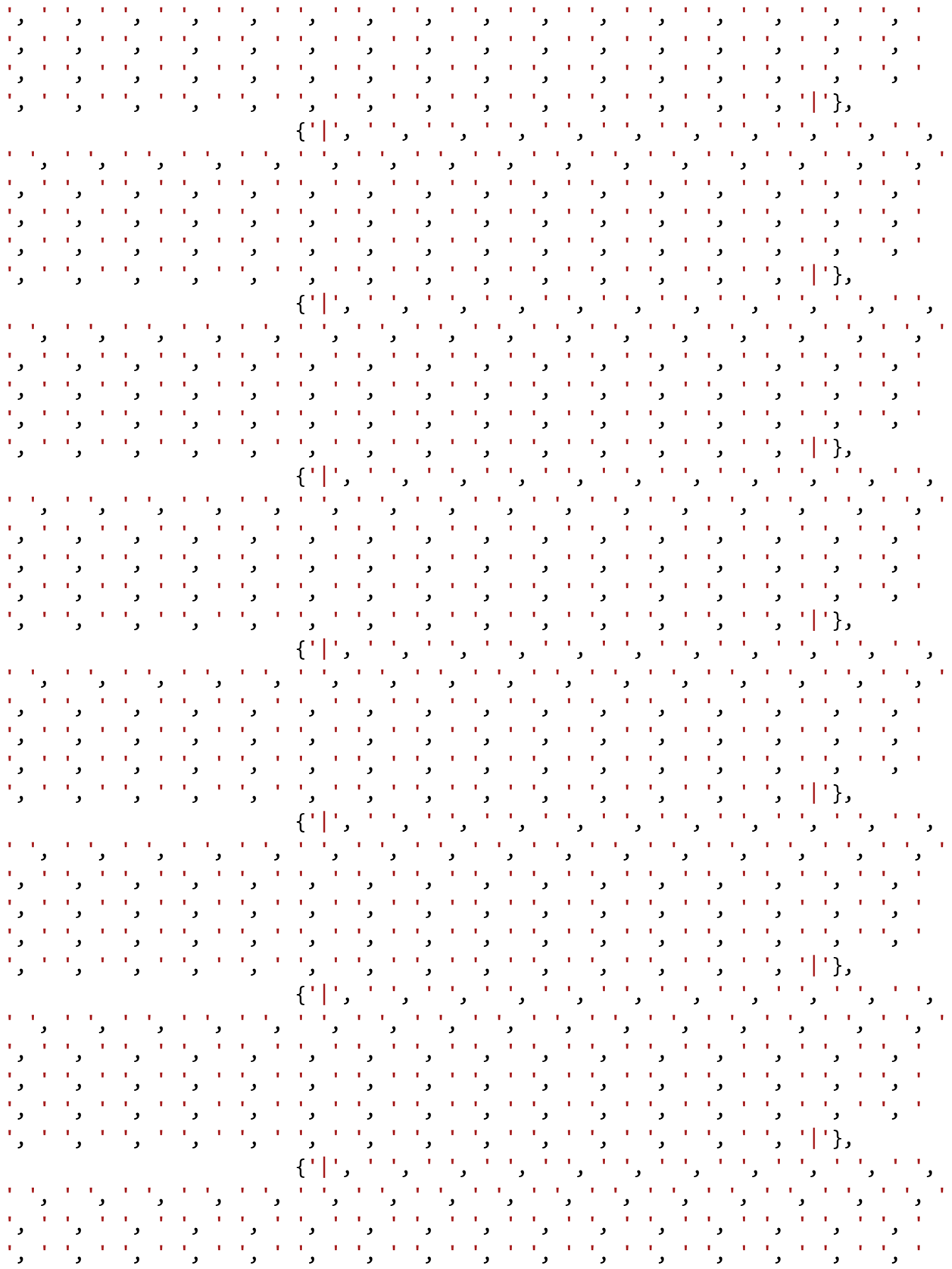
```
void bulletCollisionWithEnemy();
void bulletCollisionWithEnemy2();
void eraseEnemyAfterCollision();
void startScreen1();
void startScreen2();
void reduceEnemyLife();
void bulletCollisionWithHero();
void bulletCollisionWithHero2();
void reduceHeroLife();
void reduceEnemyLife2();
void gameover();
bool isWinner();


void gotoxy(int x, int y);
char getCharAtxy(short int x, short int y);

void gotoxy(int x, int y)
{
    COORD coordinates;
    coordinates.X = x;
    coordinates.Y = y;
    SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), coordinates);
}
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 : ' ';
}

string enemyDirection1 = "Up";
string enemyDirection2 = "Left";
int score = 0;
int option;

// Player character

char box = 219;
char heroShip[4][6] = {{' ', '|', '\\', '-', ' ', ' '},
```

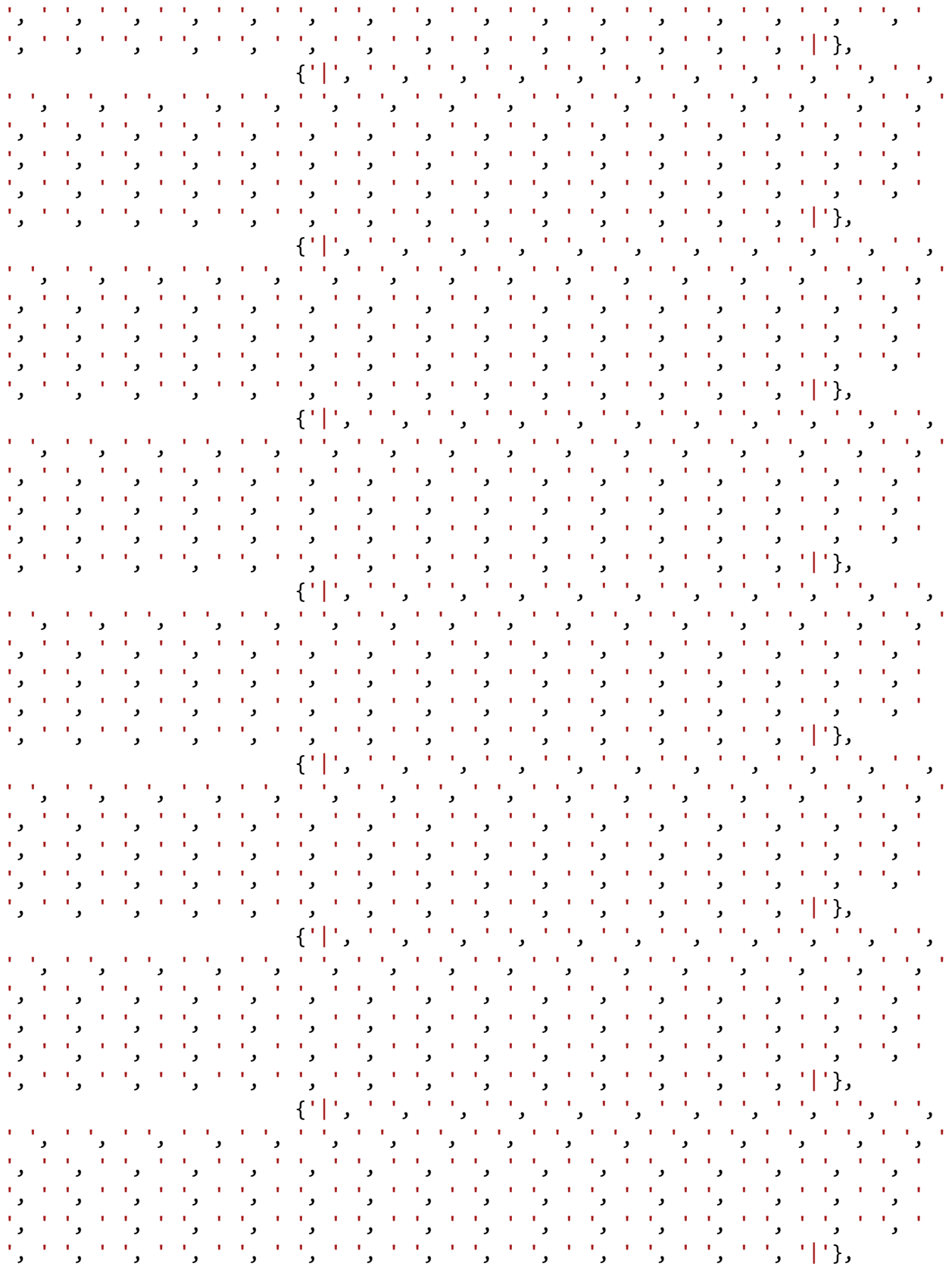
```
                        {'\\', '|', '(', ')', '=', '>'},
                        {'/', '|', '(', ')', '=', '>'},
                        {' ', '|', '/', '-', ' ', ' '}};

// Enemy character 1

char enemy1[3][6] = {{' ', ' ', ' ', ' ', ' ', '/'},
                     {'<', '=', '(', '0', ')', ' '},
                     {' ', ' ', ' ', ' ', ' ', '\\'}};

// Enemy character 2

char enemy2[3][6] = {{' ', ' ', ' ', ' ', ' ', '/'},
                     {'<', '=', '(', '0', ')', ' '},
                     {' ', ' ', ' ', ' ', ' ', '\\'}};

// Enemy character 3

char enemy3[3][6] = {{' ', ' ', ' ', ' ', ' ', '/'},
                     {'<', '=', '(', '0', ')', ' '},
                     {' ', ' ', ' ', ' ', ' ', '\\'}};

// Player coordinates
int heroShipX = 10;
int heroShipY = 5;

// Enemy coordinates 1

int enemy1X = 60;
int enemy1Y = 15;

// Enemy coordinates 2

int enemy2X = 53;
int enemy2Y = 20;

// Enemy coordinates 3

int enemy3X = 53;
int enemy3Y = 20;

// Player bullets

int bulletX[10000];
int bulletY[10000];
```

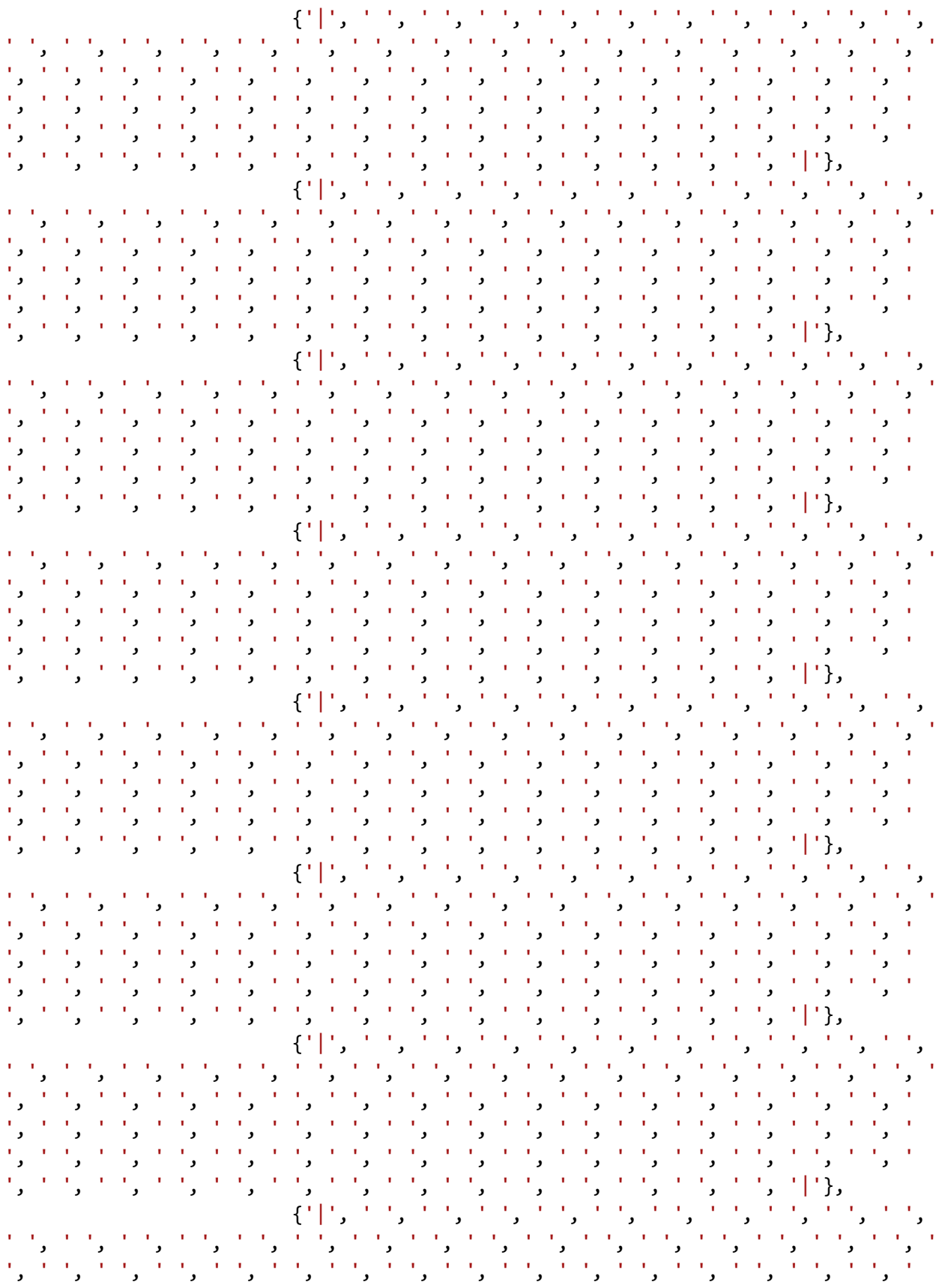
```
bool isBulletActive[10000];
int bulletCount = 0;
int timer;
// Enemy Bullets

int enemybulletX[100000];
int enemybulletY[100000];
bool isenemyBulletActive[100000];
int enemybulletCount = 0;
int enemybulletCount2 = 0;
int enemybulletCount3 = 0;

// Lives

int heroLife = 10;
int enemyLife = 5;
int enemyLife2 = 10;

main()
{
    system("cls");
    wall();

    startScreen1();
    startScreen2();
    while (true)
    {
        if (option == 1)
        {
            system("cls");
            wall();

            while (true)
            {
                if (heroLife <= 0)
                {
                    system("cls");
                    cout << "Game Over!";
                    return false;
                }
                if (GetAsyncKeyState(VK_LEFT))
                {
                    moveHeroShipLeft();
                }
                if (GetAsyncKeyState(VK_RIGHT))
```

```
{
    moveHeroShipRight();
}
if (GetAsyncKeyState(VK_UP))
{
    moveHeroShipUp();
}
if (GetAsyncKeyState(VK_DOWN))
{
    moveHeroShipDown();
}
if (GetAsyncKeyState(VK_SPACE))
{
    generateBullet();
}
if (timer == 3)
{
    moveEnemy1();
    timer = 0;
}

moveBullet();
moveenemyBullet();
bulletCollisionWithEnemy();
bulletCollisionWithHero();
bulletCollisionWithEnemy2();
bulletCollisionWithHero2();
printScore();
if (enemyLife > 0)
{
    printEnemy1();
    generateEnemyBullet();

    timer++;
}
if (enemyLife2 > 0 && enemyLife <= 0)
{
    printEnemy2();
    moveEnemy2();
    generateEnemyBullet2();
}
if (enemyLife2 <= 0)
{
    printEnemy3();
```

```
                }

                printHeroShip();

                Sleep(90);
            }
        }
        if (option = 2)
        {
            system("cls");
            wall();
            gotoxy(2, 10);
            cout << "1. Use ARROW KEYS for moving Player. You are permitted to
move the player horizontally." << endl;
            gotoxy(2, 11);
            cout << "2. Use Space Bar to fire the Player's Bullets." << endl;
            gotoxy(2, 12);
            cout << "3. Avoid the Alien's Bombs. and Debris coming towards you."
<< endl
                 << endl;
            gotoxy(2, 14);
            cout << "        *******     GOOD LUCK!   ******" << endl
                 << endl;
            gotoxy(30, 16);
            cout << "Press 1 to Play The Game";
            cin >> option;
            if (option = 1)
            {
                continue;
            }
        }
    }
}

void wall()
{
    char wall[25][90] = {{'=', '=', '=', '=', '=', '=', '=', '=', '=', '=', '=',
'=', '=', '=', '=', '=', '=', '=', '=', '=', '=', '=', '=', '=', '=', '=', '=',
'=', '=', '=', '=', '=', '=', '=', '=', '=', '=', '=', '=', '=', '=', '=', '=',
'=', '=', '=', '=', '=', '=', '=', '=', '=', '=', '=', '=', '=', '=', '=', '=',
'=', '=', '=', '=', '=', '=', '=', '=', '=', '=', '=', '=', '=', '=', '=', '=',
'=', '=', '=', '=', '=', '=', '=', '=', '=', '=', '=', '=', '=', '=', '=', '='},
                        {'|', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '
```







```
', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '
', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '
', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '|'},
                                                  {'=', '=', '=', '=', '=', '=', '=', '=', '=', '=', '=',
'=', '=', '=', '=', '=', '=', '=', '=', '=', '=', '=', '=', '=', '=', '=', '=',
'=', '=', '=', '=', '=', '=', '=', '=', '=', '=', '=', '=', '=', '=', '=', '=',
'=', '=', '=', '=', '=', '=', '=', '=', '=', '=', '=', '=', '=', '=', '=', '=',
'=', '=', '=', '=', '=', '=', '=', '=', '=', '=', '=', '=', '=', '=', '=', '=',
'=', '=', '=', '=', '=', '=', '=', '=', '=', '=', '=', '=', '=', '=', '=', '='}};
    for (int row = 0; row < 25; row++)
    {
        for (int col = 0; col < 90; col++)
        {
            cout << wall[row][col];
        }
        cout << endl;
    }
}

void printHeroShip()
{
    HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
    SetConsoleTextAttribute(hConsole, 7);
    gotoxy(heroShipX, heroShipY);
    for (int row = 0; row < 4; row++)
    {
        gotoxy(heroShipX, heroShipY + row);
        for (int col = 0; col < 6; col++)
        {
            cout << heroShip[row][col];
        }
        cout << endl;
    }
}
void printEnemy1()
{
    HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
    SetConsoleTextAttribute(hConsole, 4);
    for (int row = 0; row < 3; row++)
    {
        gotoxy(enemy1X, enemy1Y + row);
        for (int col = 0; col < 6; col++)
        {
            cout << enemy1[row][col];
        }
```

```
        cout << endl;
    }
}
void eraseEnemy1()
{
    for (int row = 0; row < 3; row++)
    {
        gotoxy(enemy1X, enemy1Y + row);
        for (int index = 0; index < 6; index++)
        {
            cout << " ";
        }
    }
}
void printEnemy2()
{
    for (int row = 0; row < 3; row++)
    {
        gotoxy(enemy2X, enemy2Y + row);
        for (int col = 0; col < 6; col++)
        {
            cout << enemy2[row][col];
        }
        cout << endl;
    }
}
void printEnemy3()
{
    for (int row = 0; row < 3; row++)
    {
        gotoxy(enemy3X, enemy3Y + row);
        for (int col = 0; col < 6; col++)
        {
            cout << enemy3[row][col];
        }
        cout << endl;
    }
}
void eraseEnemy3()
{
    for (int row = 0; row < 3; row++)
    {
        gotoxy(enemy3X, enemy3Y + row);
        for (int index = 0; index < 6; index++)
        {
```

```
            cout << " ";
        }
    }
}
void eraseEnemy2()
{
    for (int row = 0; row < 3; row++)
    {
        gotoxy(enemy2X, enemy2Y + row);
        for (int index = 0; index < 6; index++)
        {
            cout << " ";
        }
    }
}
void eraseHeroShip()
{
    for (int row = 0; row < 4; row++)
    {
        gotoxy(heroShipX, heroShipY + row);
        for (int col = 0; col < 6; col++)
        {
            cout << " ";
        }
    }
}

void moveHeroShipLeft()
{
    char next = getCharAtxy(heroShipX - 1, heroShipY);
    if (next == ' ')
    {
        eraseHeroShip();
        heroShipX = heroShipX - 1;
        printHeroShip();
    }
}
void moveHeroShipRight()
{
    char next = getCharAtxy(heroShipX + 6, heroShipY);
    if (next == ' ')
    {
        eraseHeroShip();
        heroShipX = heroShipX + 1;
        printHeroShip();
```

```
    }
}
void moveHeroShipUp()
{
    char next = getCharAtxy(heroShipX, heroShipY - 1);
    if (next == ' ')
    {
        eraseHeroShip();
        heroShipY = heroShipY - 1;
        printHeroShip();
    }
}
void moveHeroShipDown()
{
    char next = getCharAtxy(heroShipX, heroShipY + 4);
    if (next == ' ')
    {
        eraseHeroShip();
        heroShipY = heroShipY + 1;
        printHeroShip();
    }
}
void generateBullet()
{
    bulletX[bulletCount] = heroShipX + 7;
    bulletY[bulletCount] = heroShipY + 2;
    isBulletActive[bulletCount] = true;
    gotoxy(heroShipX + 7, heroShipY + 2);
    cout << "*";
    bulletCount++;
}
void generateEnemyBullet()
{
    enemybulletX[enemybulletCount] = enemy1X - 1;
    enemybulletY[enemybulletCount] = enemy1Y + 1;
    isenemyBulletActive[enemybulletCount] = true;
    gotoxy(enemy1X - 1, enemy1Y + 1);
    cout << ".";
    enemybulletCount++;
}
void generateEnemyBullet2()
{
    enemybulletX[enemybulletCount2] = enemy2X - 1;
    enemybulletY[enemybulletCount2] = enemy2Y + 1;
    isenemyBulletActive[enemybulletCount2] = true;
```

```
    gotoxy(enemy2X - 1, enemy2Y + 1);
    cout << ".";
    enemybulletCount2++;
}
void moveBullet()
{
    for (int x = 0; x < bulletCount; x++)
    {
        if (isBulletActive[x] == true)
        {
            char next = getCharAtxy(bulletX[x] + 1, bulletY[x]);
            if (next != ' ')
            {
                eraseBullet(bulletX[x], bulletY[x]);
                makeBulletInactive(x);
            }

            else
            {
                eraseBullet(bulletX[x], bulletY[x]);
                bulletX[x] = bulletX[x] + 1;
                printBullet(bulletX[x], bulletY[x]);
            }
        }
    }
}
void moveenemyBullet()
{
    for (int x = enemybulletCount; x > 0; x--)
    {
        if (isenemyBulletActive[x] == true)
        {
            char next = getCharAtxy(enemybulletX[x] - 1, enemybulletY[x]);
            if (next != ' ')
            {
                eraseenemyBullet(enemybulletX[x], enemybulletY[x]);
                makeenemyBulletInactive(x);
            }
            else
            {
                eraseenemyBullet(enemybulletX[x], enemybulletY[x]);
                enemybulletX[x] = enemybulletX[x] - 1;
                printenemyBullet(enemybulletX[x], enemybulletY[x]);
            }
        }
```

```
    }
}
void printBullet(int x, int y)
{
    HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
    SetConsoleTextAttribute(hConsole, 7);
    gotoxy(x, y);
    cout << "*";
}
void printenemyBullet(int x, int y)
{
    HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
    SetConsoleTextAttribute(hConsole, 6);
    gotoxy(x, y);
    cout << ".";
}
void eraseBullet(int x, int y)
{
    gotoxy(x, y);
    cout << " ";
}
void eraseenemyBullet(int x, int y)
{
    gotoxy(x, y);
    cout << " ";
}
void makeBulletInactive(int index)
{
    isBulletActive[index] = false;
}
void makeenemyBulletInactive(int index)
{
    isenemyBulletActive[index] = false;
}
void moveEnemy1()
{

    if (enemyDirection1 == "Up")
    {
        char next = getCharAtxy(enemy1X, enemy1Y - 1);
        if (next == ' ')
        {
            eraseEnemy1();
            enemy1Y--;
            printEnemy1();
```

```
        }
        if (next == '=')
        {
            enemyDirection1 = "Down";
        }
    }
    if (enemyDirection1 == "Down")
    {
        char next = getCharAtxy(enemy1X, enemy1Y + 3);
        if (next == ' ')
        {
            eraseEnemy1();
            enemy1Y++;
            printEnemy1();
        }
        if (next == '=')
        {
            enemyDirection1 = "Up";
        }
    }
}
void moveEnemy2()
{
    if (enemyDirection1 == "Up")
    {
        char next = getCharAtxy(enemy2X, enemy2Y - 1);
        if (next == ' ')
        {
            eraseEnemy2();
            enemy2Y--;
            printEnemy2();
        }
        if (next == '=')
        {
            enemyDirection1 = "Down";
        }
    }
    if (enemyDirection1 == "Down")
    {
        char next = getCharAtxy(enemy2X, enemy2Y + 3);
        if (next == ' ')
        {
            eraseEnemy2();
            enemy2Y++;
            printEnemy2();
```

```
        }
        if (next == '=')
        {
            enemyDirection1 = "Up";
        }
    }
}

void bulletCollisionWithEnemy()
{
    for (int x = 0; x < bulletCount; x++)
    {
        if (enemyLife > 0)
        {
            if (isBulletActive[x] == true)
            {
                if (bulletX[x] + 1 == enemy1X && (bulletY[x] == enemy1Y ||
bulletY[x] == enemy1Y + 2 || bulletY[x] == enemy1Y + 3))
                {
                    addScore();
                    reduceEnemyLife();
                }
                if (enemy1X - 1 == bulletX[x] && enemy1Y + 1 == bulletY[x])
                {
                    addScore();
                    reduceEnemyLife();
                }
                if (enemyLife <= 0)
                {
                    eraseEnemy1();
                    printEnemy2();
                }
            }
        }
    }
}
void bulletCollisionWithEnemy2()
{
    for (int x = 0; x < bulletCount; x++)
    {
        if (enemyLife2 > 0 && enemyLife<=0)
        {
            if (isBulletActive[x] == true)
            {
```

```
                if (bulletX[x] + 1 == enemy2X && (bulletY[x] == enemy2Y ||
bulletY[x] == enemy2Y + 2 || bulletY[x] == enemy2Y + 3))
                {
                    addScore();
                    reduceEnemyLife2();
                }
                if (enemy2X - 1 == bulletX[x] && enemy2Y + 1 == bulletY[x])
                {
                    addScore();
                    reduceEnemyLife2();
                }
                if (enemyLife2 <= 0)
                {
                    eraseEnemy2();
                    printEnemy3();
                }
            }
        }
    }
}
void bulletCollisionWithHero()
{
    for (int x = 0; x < enemybulletCount; x++)
    {
        if (isenemyBulletActive[x] == true)
        {
            if (enemybulletX[x] == heroShipX + 5 && enemybulletY[x] == heroShipY)
            {
                reduceHeroLife();
            }
            if (heroShipX - 1 == enemybulletX[x] && heroShipY + 1 ==
enemybulletY[x])
            {
                reduceHeroLife();
            }
        }
    }
}
void bulletCollisionWithHero2()
{
    for (int x = 0; x < enemybulletCount2; x++)
    {
        if (isenemyBulletActive[x] == true)
        {
            if (enemybulletX[x] == heroShipX + 5 && enemybulletY[x] == heroShipY)
```

```
            {
                reduceHeroLife();
            }
            if (heroShipX - 1 == enemybulletX[x] && heroShipY + 1 ==
enemybulletY[x])
            {
                reduceHeroLife();
            }
        }
    }
}
void addScore()
{
    score++;
}
void reduceEnemyLife()
{
    enemyLife--;
}
void reduceEnemyLife2()
{
    enemyLife2--;
}
void reduceHeroLife()
{
    gotoxy(3, 17);
    heroLife--;
}
void printScore()
{
    HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
    SetConsoleTextAttribute(hConsole, 11);
    gotoxy(100, 5);
    cout << "Score:" << score << '\t';
    cout << "Enemy Life:" << enemyLife << '\t';
    cout << "Enemy Life 2:" << enemyLife2 <<" "<< '\t';
    cout << "Hero Life:" << heroLife << " ";
}

void startScreen1()
{
    HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
    SetConsoleTextAttribute(hConsole, 30);
    gotoxy(20, 4);
```

```
    cout << box << box << box << box << box << box << box << box << box << "   "
<< box << box << box << box << box << box << box << box << box << box << "    "
<< box << box << "    " << box << box << box << box << box << box << box << box
<< box << "  " << box << box << box << box << box << box << box << box << box <<
endl;
    gotoxy(20, 5);
    cout << box << box << box << "         " << box << box << box << "    " <<
box << box << box << "  " << box << box << "  " << box << box << "  " << box <<
box << box << "        " << box << box << box << endl;
    gotoxy(20, 6);
    cout << box << box << box << box << box << box << box << box << box << "   "
<< box << box << box << box << box << box << box << box << box << box << " " <<
box << box << "    " << box << box << " " << box << box << box << "        " <<
box << box << box << box << box << box << box << box << endl;
    gotoxy(20, 7);
    cout << "      " << box << box << box << "   " << box << box << box <<
"        " << box << box << box << box << box << box << box << box << " " << box
<< box << box << "        " << box << box << box << endl;
    gotoxy(20, 8);
    cout << box << box << box << box << box << box << box << box << box << "   "
<< box << box << box << "        " << box << box << box << "  " << box << box <<
box << " " << box << box << box << box << box << box << box << box << box << "  "
<< box << box << box << box << box << box << box << box << box << endl;

    cout << endl;
}
void startScreen2()
{
    gotoxy(7, 10);
    cout << box << box << box << box << box << box << box << box << box << "   "
<< box << box << box << "    " << box << box << box << "  " << box << box << box
<< box << box << box << box << "  " << box << box << box << box << box << box <<
box << "  " << box << box << box << box << box << box << box << box << box << box
<< "  " << box << box << box << box << box << box << box << box << box << "  " <<
box << box << box << box << box << box << box << box << box << endl;
    gotoxy(7, 11);
    cout << box << box << box << "         " << box << box << box << "    " <<
box << box << box << "  " << box << "     " << box << "  " << box << "     " <<
box << "      " << box << box << "      " << box << box << box << "        " <<
box << box << box << "   " << box << box << box << endl;
    gotoxy(7, 12);
    cout << box << box << box << box << box << box << box << box << box << "   "
<< box << box << box << box << box << box << box << box << box << box << "  " <<
box << "     " << box << "  " << box << "     " << box << "      " << box << box
```

```
<< "      " << box << box << box << box << box << box << box << "   " <<
box << box << box << box << box << box << box << box << endl;
    gotoxy(7, 13);
    cout << "      " << box << box << box << "   " << box << box << box << "    "
<< box << box << box << "  " << box << "     " << box << "  " << box << "     "
<< box << "      " << box << box << "      " << box << box << box << "        "
<< box << box << box << "" << box << box << endl;
    gotoxy(7, 14);
    cout << box << box << box << box << box << box << box << box << box << "   "
<< box << box << box << "    " << box << box << box << "  " << box << box << box
<< box << box << box << box << "  " << box << box << box << box << box << box <<
box << "      " << box << box << "      " << box << box << box << box << box <<
box << box << box << box << "  " << box << box << box << "  " << box << box <<
box << endl;
    gotoxy(7, 15);
    cout << endl
         << endl
         << endl
         << endl
         << endl;
    gotoxy(35, 18);
    HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
    SetConsoleTextAttribute(hConsole, 10);
    cout << "Press 1 to play" << endl;
    gotoxy(35, 20);
    cout << "Press 2 to instructions";
    cin >> option;
}
void gameover()
{
    system("cls");
    cout << endl;
    cout << "\t\t--------------------------" << endl;
    cout << "\t\t-------- Game Over -------" << endl;
    cout << "\t\t--------------------------" << endl
         << endl;
    cout << "\t\tPress any key to go back to menu.";
    getch();
}
```