Tower Wars

Session: 2022 – 2026

Submitted by:

Mohammad Abdullah 2022-CS-155

Supervised by:

Dr. Awais Hassan

Department of Computer Science

University of Engineering and Technology, Lahore, Pakistan

Table of Contents

- # Description and Story of Game

A city is in danger. The city has been attacked by the monstrous forces of the enemies. The army of the city has been defeated. The citizens of the city are discouraged and embarrassed. Now the peace, joy, prosperity and integrity of the city and its citizen rests on the shoulder of the Rambo.

Now the responsibility of carrying the cities defence have been handed to the Rambo. He would have to deal with the advancing enemies and remove them to reinstate the lost lights and hopes of the people.

If the player succeeds in defeating the enemies, he will be able to restore the kingdom's peace and integrity. But if he fails to do that, the kingdom will fall, and so will be its citizens.

- # Game Characters Description
  - ## Player

There is one human player in the Game.

Rambo:
Rambo is a patriotic soldier who have strong self-belief that he can single-handedly defeat a strong army. He is adventurous and loves to take on challenges. He also has a tactical mind. He thinks that he can still defeat the enemy although the enemies have outnumbered him.

  - ## Enemies

There are 4 enemies in the game.

Infantry:
Infantry is the weakest force among the enemies. It does not cause much damage and are also defeated 1 bullet. But they do have an advantage. Although they are the weakest but they are in great numbers and they do cause difficulty to Rambo.

Tanks:
Tanks are an important part for the opponent's army. They are not in great numbers. But they do cause a lot of damage. Tanks are also slow in pace. They are also not defeated as easily as infantry. They need at least 5 bullets to be defeated.

Fighter Jets:
Fighter Jets provide great support to their army. They do cause a lot of damage to the kingdom's wall. Their assistance is vital for their army but not for Rambo and its city. It takes them 10 shots to the planes to put them down.

Bala:
Bala is the leader of the opponent's army. He comes at the end when all other

resources of their army are finished. He is the strongest of his army and he uses his strength at the end. It will take 25 bullets to finish his health.

• Game Objects Description

Following are the Objects in the Game

○ Power Pill:

A Power Pill is an energizer that would increase the health of Kingdom's wall. It will increase its health by 10.

○ Tower Walls:

Tower Walls are the walls that should be protected by Rambo from the enemies.

• Rules & Interactions

Rambo can eat Power pills to increase the health of the wall so that it can withstand more. Rambo also have to kill all the enemies so that he can restore the peace of the city. Rambo have only three towers to protect the kingdom so he will have to protect the kingdom until the last tower is left. If the last tower is also destroyed by the enemy, the game would be lost. The game can be won only if Rambo kills the Bala.

1. Rambo can eat Power pills to increase health of the walls.
2. Rambo will have to kill all the enemies to win.
3. Collision with any enemy would result in defeat.
4. Health of Tower is 300.
5. Health of Infantry is 5.
6. Health of Tank is 10.
7. Health of Jet Plane is 15.
8. Health of Bala is 20.

• Goal of the Game

The goal of the game is to remove all the enemies of the opponent army and kill their leader Bala while having some part of the kingdom's wall.

- **Wireframes**

Figure 1: Welcome Menu

Figure 2: Main Menu

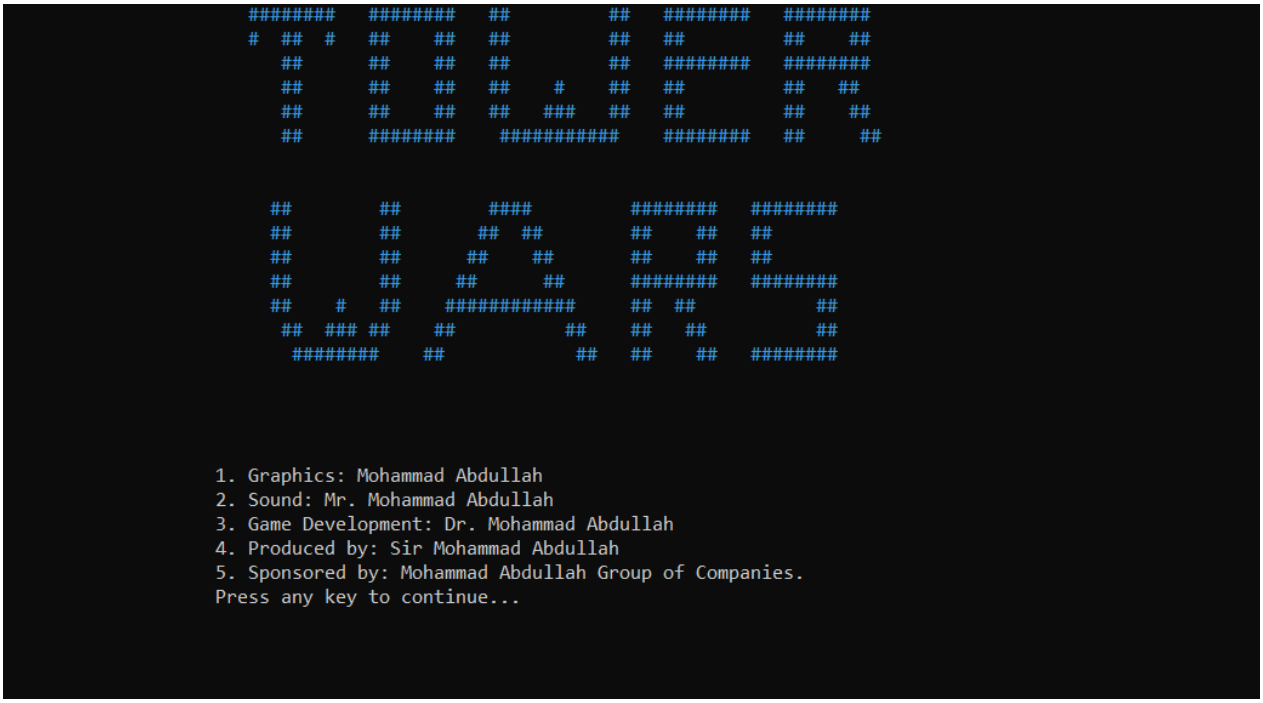

Figure 3: Credits Menu

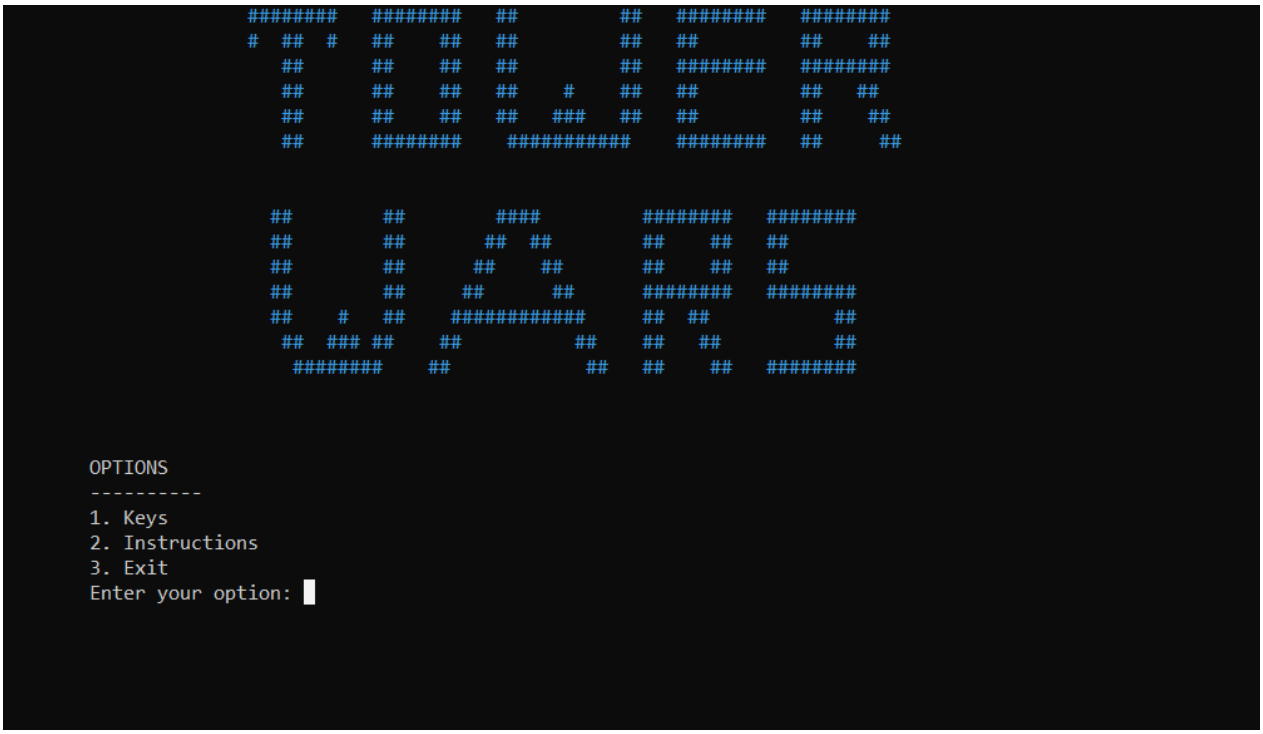

Figure 4: Options Menu

Figure 5: Keys

Figure 6: Instructions

Figure 7: Game

- **Data Structures**
  - char backslash = 92;
  - char box = 219;
  - char fire = 170;
  - char rambo[4][9] = {
    {'[', '=', '=', '=', '=', '=', '=', '=', backslash},
    {'|', '=', '=', '=', '=', '=', '=', '=', '/'},
    {'|', '|', ' ', '>', ' ', ' ', ' ', ' ', ' '},
    {'|', '|', ' ', ' ', ' ', ' ', ' ', ' ', ' '}}
  - char infantry[3][11] = {{' ', ' ', ' ', ' ', ' ', '(', '-', ' ', '-', ')', ' '},
    {'<', '-', '-', fire, backslash, box, box, box, box, box, '/'},
    {' ', ' ', ' ', ' ', ' ', '|', ' ', ' ', ' ', '|', ' '}}
  - char tank[3][5] = {{'<', '-', '-', '-', fire},
    {box, box, box, box, box},
    {'O', 'O', 'O', 'O', 'O'}}
  - char jetPlane[5][6] = {{' ', ' ', ' ', ' ', '/', '|'},
    {' ', '/', backslash, '/', ' ', '|'},
    {'<', '[', ' ', ' ', ' ', '|'},
    {' ', backslash, '/', backslash, ' ', '|'},
    {' ', ' ', ' ', ' ', backslash, '|'}}
  - char bala[4][9] = {
    {'/', '=', '=', '=', '=', '=', '=', '=', ']'},
    {backslash, '=', '=', '=', '=', '=', '=', '=', '|'},
    {' ', ' ', ' ', ' ', ' ', '<', ' ', '|', '|'},
    {' ', ' ', ' ', ' ', ' ', ' ', ' ', '|', '|'}}
  - int bulletX[500];
  - int bulletY[500];
  - bool isBulletActive[500];
  - int enemyBulletX[500];
  - int enemyBulletY[500];
  - bool isEnemyBulletActive[500];

- **Function Prototypes**
  - void gotoxy(int x, int y);
  - char getCharAtxy(short int x, short int y);
  - void header();
  - void welcome();
  - string mainMenu();
  - string optionSubMenu();
  - float distance(int ramboX, int ramboY);
  - void gameInterface();
  - void optionsInterface();
  - void creditsInterface();
  - void showInstructions();
  - void showKeys();
  - void printMaze();
  - void printInfantry();
  - void printRambo();
  - void removeRambo();
  - void moveRamboUp();
  - void moveRamboDown();
  - void moveRamboLeft();
  - void moveRamboRight();
  - void generateBullet();
  - void printBullet(int x, int y);
  - void eraseBullet(int x, int y);
  - void moveBullet();
  - void printBullet(int x, int y);
  - void eraseBullet(int x, int y);
  - void makeBulletInactive(int idx);
  - void addScore();
  - void printScore();
  - void moveInfantry();
  - void printInfantry();
  - void eraseInfantry();
  - void bulletCollisionWithInfantry();
  - void moveTank();
  - void printTank();
  - void eraseTank();
  - void bulletCollisionWithTank();
  - void printTankHealth();
  - void moveJetPlane();

- void printJetPlane();
- void eraseJetPlane();
- void bulletCollisionWithJetPlane();
- void printJetPlaneHealth();
- void addScore();
- void printScore();
- void printInfantryHealth();
- void moveBalaRandom();
- void moveBalaSmart();
- void printBala();
- void eraseBala();
- void bulletCollisionWithBala();
- void printBalaHealth();
- void generateEnemyBullet(int x, int y);
- void moveEnemyBullet();
- void makeEnemyBulletInactive(int idx);
- void printEnemyBullet(int x, int y);
- void eraseEnemyBullet(int x, int y);
- void printHealthPill();
- void removeHealthPill();
- void healthPill();
- void bulletCollisionWithTower();
- bool playerCollisionWithEnemyRight();
- bool playerCollisionWithEnemyLeft();
- bool playerCollisionWithEnemyUp();
- bool playerCollisionWithEnemyDown();
- void printRamboHealth();
- void gameOver();
- void youWon();
- float distance(int ramboX, int ramboY);
- void nextLevel();
- bool setCursor(bool visible);

- # Complete Code

```
main()
{
setCursor(0);
 string option = "7";
 bool file = true;
 system("cls");
 welcome();
 while (file == true)
 {
  system("cls");
  header();
  option = mainMenu();
  // Takes user to the game
  if (option == "1")
  {
   gameInterface();
  }
  // Shows options to the user
  if (option == "2")
  {
   optionsInterface();
  }
  // Shows Credits
  if (option == "3")
  {
   creditsInterface();
  }
  // Terminates Program
  if (option == "4")
  {
   file = false;
  }
 }
}
```

```
void gotoxy(int x, int y)
 {
  COORD coordinates;
  coordinates.X = x;
  coordinates.Y = y;
 SetConsoleCursorPosition(GetStdHandle(STD_OU
 TPUT_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 : ' ';
 }
 void header()
 {
  HANDLE hConsole =
 GetStdHandle(STD_OUTPUT_HANDLE);
  SetConsoleTextAttribute(hConsole, 3);
  cout << "                ########  ########  ##
 ##  ########  ######## " << endl;
 cout << "               # ## #  ##   ##  ##      ##
 ##       ##   ## " << endl;
 cout << "                 ##    ##   ##  ##      ##
 ########  ######## " << endl;
  cout << "                 ##    ##   ##  ##   #   ##
 ##       ##  ##  " << endl;
```

```
 cout << "                  ##    ##   ##  ##  ###
##  ##      ##   ##  " << endl;
 cout << "                  ##    ########
###########   ########  ##    ## " << endl;
 cout << endl
    << endl;

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

 cout << endl
    << endl
    << endl;
}
void welcome()
{
 // Changes Text Colour to Yellow
 HANDLE hConsole =
GetStdHandle(STD_OUTPUT_HANDLE);
 SetConsoleTextAttribute(hConsole, 6);
 cout << endl;
cout << "               __     __       .__
" << endl;
 cout << "              / \\   / \\ ____  | |   ____
____   _____   ____  " << endl;
cout << "              \\  \\/\\/  /_/ __ \\ | |  _/
___\\ /  _ \\ /    \\ _/ __ \\" << endl;
cout << "               \\       / \\ ___/ |  |__\\
\\___ (  <_> )|  Y Y  \\\\  ___/ " << endl;
 cout << "                \\__/\\ /  \\___  >|____/
\\___  > \\____/ |__|_|  / \\___  >" << endl;
 cout << "                     \\/       \\/         \\/
\\/     \\/" << endl;
 cout << endl
    << endl
    << endl;
 SetConsoleTextAttribute(hConsole, 10);
 // Changes Text Colour to Green
 cout << "                                -|           |-
" << endl;
 cout << "              -|              [-_-_-_-_-_-_-_-]
|-           " << endl;
 cout << "              [-_-_-_-_-]        |          |
[-_-_-_-_-]          " << endl;
 cout << "               | o   o |         [  0   0   0  ]
| o   o |           " << endl;
 cout << "               |    |   -|      |         |      |-
|    |           " << endl;
 cout << "               |    |_-___-___-___-|
|-___-___-___-_|    |           " << endl;
 cout << "               |  o  ]           [    0    ]
[  o  |           " << endl;
cout << "               |    ]   o   o   o [ _______ ]  o
o   o   [    | ----_____________" << endl;
 cout << "        _____----- |    ]          [ |||||||| ]
[    |           " << endl;
 cout << "               |    ]          [ |||||||| ]
[    |           " << endl;
 cout << "             _-_-|_____]----------------
[_|||||||_]--------------[_____|-_-_         " <<
endl;
 cout << "            ( (____________------------
_________________----------_________) )        " <<
endl;

 string a;
```

```
 cout << "Press any key to continue...";
 getline(cin, a);
}
string mainMenu()
{
 // Changes Text Colour to White
 HANDLE hConsole =
GetStdHandle(STD_OUTPUT_HANDLE);
 SetConsoleTextAttribute(hConsole, 7);
 // Prints Main Menu
 cout << "       MENU" << endl;
 cout << "       ----------" << endl;
 cout << "       1. Start" << endl;
 cout << "       2. Options" << endl;
 cout << "       3. Credits" << endl;
 cout << "       4. Exit" << endl;
 string option;
 cout << "       Enter your option: ";
 getline(cin, option); // Takes an option from
user
 return option;
}
string optionSubMenu()
{
 // Changes Text Colour to White
 HANDLE hConsole =
GetStdHandle(STD_OUTPUT_HANDLE);
 SetConsoleTextAttribute(hConsole, 7);
 // Prints Options Menu
 cout << "       OPTIONS" << endl;
 cout << "       ----------" << endl;
 cout << "       1. Keys" << endl;
 cout << "       2. Instructions" << endl;
 cout << "       3. Exit" << endl;
 string option;
 cout << "       Enter your option: ";
 getline(cin, option); // Takes an option from
user
 return option;
}
```

```
void gameInterface(){
  // Reinitiate all the valuse so the game could
 run the same way it started
  int healthTimer = 0;
  int smartTimer = 0;
  int level = 1;
  towerHealth = 300;
  balaHealth = 20;
  infantryHealth = 5;
  tankHealth = 10;
  jetPlaneHealth = 15;
  ramboX = 15;
  ramboY = 10;
  infantryX = 60;
  infantryY = 15;
  tankX = 90;
  tankY = 7;
  jetPlaneX = 80;
  jetPlaneY = 7;
  balaX = 60;
  balaY = 15;
  enemy = 4;
  enemyFireDamage = 5;
  // boolean values to check collision
  bool collisionPlayerRight, collisionPlayerLeft,
collisionPlayerUp, collisionPlayerDown;
  bool gameRunning = true;
  system("cls");
  printMaze();
  printRambo();
printInfantry();
  printRamboHealth(); // Prints Rambos Health
  while (gameRunning == true)
  {
   if (towerHealth > 0 && balaHealth > 0)
   {
    if (GetAsyncKeyState(VK_LEFT))
    {
     moveRamboLeft(); // Moves Player to the Left
    }
```

```
if (GetAsyncKeyState(VK_RIGHT))
  {
   moveRamboRight(); // Moves Player to the
Right
  }

  if (GetAsyncKeyState(0x41))
  {
   moveRamboUp(); // Moves Player to the Up
  }

  if (GetAsyncKeyState(VK_DOWN))
  {
   moveRamboDown(); // Moves Player to the
Down
  }
  if (GetAsyncKeyState(VK_SPACE))
  {
   generateBullet(); // Fires
  }
  if (GetAsyncKeyState(VK_ESCAPE))
  {
   gameRunning = false; // Quits the game
  }
  if (enemy > 0)
  {
   if (enemy == 4) // Infantry
   {
    enemyFireDamage = 1;
    moveInfantry();
    bulletCollisionWithInfantry();
    printInfantryHealth();
    if (timer == 16)
    {
     generateEnemyBullet(infantryX, infantryY +
1);
     timer = 0;
    }
    moveEnemyBullet();
    timer++;
if (infantryHealth <= 0)
    {
     enemy--;
     timer = 0;
    }
   }
   if (enemy == 3) // Tanks
   {
    enemyFireDamage = 2;
    moveTank();
    bulletCollisionWithTank();
    printTankHealth();
    if (timer == 12)
    {
     generateEnemyBullet(tankX, tankY);
     timer = 0;
    }

    moveEnemyBullet();
    timer++;
    if (tankHealth <= 0)
    {
     enemy--;
     timer = 0;
    }
   }
   if (enemy == 2) // Jet Plane
   {
    enemyFireDamage = 3;
    moveJetPlane();
    bulletCollisionWithJetPlane();
    printJetPlaneHealth();
    if (timer == 8)
    {
     generateEnemyBullet(jetPlaneX - 1, jetPlaneY
+ 2);
     timer = 0;
    }
```

```
moveEnemyBullet();
    timer++;
    if (jetPlaneHealth <= 0)
    {
     enemy--;
     timer = 0;
    }
   }
   if (enemy == 1) // Final Enemy
   {
    if (level == 1)
    {
     nextLevel();
     level = 0;
    }
    enemyFireDamage = 4;
    moveBalaRandom();
    bulletCollisionWithBala();
    printBalaHealth();
    if (smartTimer == 0)
    {
     moveBalaSmart();
     smartTimer = 0;
    }
    smartTimer++;
    if (timer == 4)
    {
     generateEnemyBullet(balaX - 1, balaY);
     timer = 0;
    }
    moveEnemyBullet();
    timer++;
    if (balaHealth <= 0)
    {
     enemy--;
     timer = 0
```

```
}
    }
   }
   // Health Pill Comes into the game
   if (healthTimer == 50)
   {
    healthPill();
    healthTimer = 0;
   }
   bulletCollisionWithTower();
   moveBullet();
   printScore();
   printRamboHealth();
   // Checking Collisions
   collisionPlayerRight =
playerCollisionWithEnemyRight();
   collisionPlayerLeft =
playerCollisionWithEnemyLeft();
   collisionPlayerUp =
playerCollisionWithEnemyUp();
   collisionPlayerDown =
playerCollisionWithEnemyDown();
   if ((collisionPlayerLeft == true) ||
(collisionPlayerRight == true) ||
(collisionPlayerUp == true) ||
(collisionPlayerDown == true))
   {
    gameOver();
    gameRunning = false;
   }
   // Checking if the player has lost
   if (towerHealth <= 0)
   {
    gameOver();
    gameRunning = false;
   }
   // Timer for Increasing health system
   healthTimer++;
  }
 }
```

```
// Checking if the player has Won
   if (balaHealth <= 0)
   {
    youWon();
    gameRunning = false;
   }
   Sleep(30);
   // Timer for Increasing health system
   healthTimer++;
  }
 }
}

void optionsInterface()
{
 string option = "1";
 while (option != "3")
 {
  system("cls");
  header();              // Prints header
  option = optionSubMenu(); // Takes option
from the user
  // Shows keys for the game
  if (option == "1")
  {
   showKeys();
  }

  // Instructions for the user
  if (option == "2")
  {
   showInstructions();
  }
 }
}
```

```
void creditsInterface()
{
 system("cls");
 header();
 gotoxy(20, 19);
 HANDLE hConsole =
GetStdHandle(STD_OUTPUT_HANDLE);
 SetConsoleTextAttribute(hConsole, 7); //
Changes Colour to White Text and Black BG
 Sleep(90);
 cout << "1. Graphics: Mohammad Abdullah";
 Sleep(90);
 gotoxy(20, 20);
 cout << "2. Sound: Mr. Mohammad Abdullah ";
 Sleep(90);
 gotoxy(20, 21);
 cout << "3. Game Development: Dr. Mohammad
Abdullah";
 Sleep(90);
 gotoxy(20, 22);
 cout << "4. Produced by: Sir Mohammad
Abdullah";
 Sleep(90);
 gotoxy(20, 23);
 cout << "5. Sponsored by: Mohammad Abdullah
Group of Companies.";
 gotoxy(20, 24);
 Sleep(90);
 cout << "Press any key to continue...";
 // Any string to continue the game later
 string a;
 getline(cin, a);
}
```

```
void showInstructions()
{
 system("cls");
 header();
 HANDLE hConsole =
GetStdHandle(STD_OUTPUT_HANDLE);
 SetConsoleTextAttribute(hConsole, 7); //
Changes Colour to White Text and Black BG
 // Shows Instructions
 cout << "        1. The player can move in all the
directions." << endl;
 cout << "        2. The player will have to kill all
the enemies to win." << endl;
 cout << "        Press any key to exit... " << endl;
 // Any string to continue the game later
 string a;
 getline(cin, a);
}

void showKeys()
{
 system("cls");
 header();
 HANDLE hConsole =
GetStdHandle(STD_OUTPUT_HANDLE);
 SetConsoleTextAttribute(hConsole, 7); //
Changes Colour to White Text and Black BG
 // Shows Keys
 cout << "   Up       Key Up" << endl;
 cout << "  Down      Key Down" << endl;
 cout << "  Left      Key Left" << endl;
 cout << "  Right     Key Right" << endl;
 cout << "  Fire      Key Space" << endl;
 cout << "  Exit      Key Escape" << endl;
 cout << "Enter any key to exit...";
 // Any string to continue the game later
 string a;
 getline(cin, a);
}
```

```
void printMaze()
{
 HANDLE hConsole =
GetStdHandle(STD_OUTPUT_HANDLE);
 SetConsoleTextAttribute(hConsole, 5); //
Changes Text Colour of Maze to Purple
 string maze[100];
 int idx = 0;
 fstream file;
 string line;
 file.open("Maze.txt", ios::in);
 while (!file.eof())
 {
  getline(file, line);
  maze[idx] = line;
  idx++;
 }
 file.close();

 for (int x = 0; x < idx; x++)
 {
  cout << maze[x];
  cout << endl;
 }
}

void printTank()
{
 // Changes Text Colour of Maze to Red
 HANDLE hConsole =
GetStdHandle(STD_OUTPUT_HANDLE);
 SetConsoleTextAttribute(hConsole, 4);

 // Tank Characters
 char box = 219;
 char fire = 170;
 char tank[3][5] = {{'<', '-', '-', '-', fire}, {box, box,
box, box, box}, {'O', 'O', 'O', 'O', 'O'}};
 }
```

```
 // Prints Tank
 for (int i = 0; i < 3; i++)
 {
  gotoxy(tankX, tankY + i); // Goes below to print next line
  for (int j = 0; j < 5; j++)
  {
   cout << tank[i][j];
  }
 }
}

void eraseTank()
{
 // Erase the tank
 gotoxy(tankX, tankY);
 for (int x = 0; x < 5; x++)
 {
  cout << " ";
 }
 gotoxy(tankX, tankY + 1);
 for (int x = 0; x < 5; x++)
 {
  cout << " ";
 }
 gotoxy(tankX, tankY + 2);
 for (int x = 0; x < 5; x++)
 {
  cout << " ";
 }
}
void bulletCollisionWithTank()
{
 for (int x = 0; x < bulletCount; x++)
 {
  if (isBulletActive[x] == true)
  {
   if (bulletX[x] + 1 == tankX && (bulletY[x] == tankY || bulletY[x] == tankY + 1 || bulletY[x] == tankY + 2))
{
    addScore();
    tankHealth--;
   }
   else if (tankX - 1 == bulletX[x] && tankY + 1 == bulletY[x])
   {
    addScore();
    tankHealth--;
   }}}}
void printTankHealth()
{
 char heart = 3;
 // Prints Tanks Health at left side of maze
 gotoxy(120, 3);
 if (tankHealth <= 10 && tankHealth > 8)
 {
  cout << "Tank: " << heart << heart << heart << heart << heart;
 }
 gotoxy(120, 3);
 if (tankHealth <= 8 && tankHealth > 6)
 {
  cout << "Tank: " << heart << heart << heart << heart << " ";
 }
 gotoxy(120, 3);
 if (tankHealth <= 6 && tankHealth > 4)
 {
  cout << "Tank: " << heart << heart << heart << " "
      << " ";
 }
 gotoxy(120, 3);
 if (tankHealth <= 4 && tankHealth > 2)
 {
  cout << "Tank: " << heart << heart << " "
      << " " << " ";
 }
 gotoxy(120, 3);
 if (tankHealth <= 2 && tankHealth > 0)
 {
cout << "Tank: " << heart << " "  << " "  << " " << " “; }
```

```
gotoxy(120, 3);
 if (tankHealth <= 0)
 {
  cout << "     "   << " "  << " "  << " " << " "  << " ";
  gotoxy(tankX, tankY);
  eraseTank();
 }
}
void moveJetPlane()
{
 // Jet Plane Moving Functionality
 if (jetPlaneDirection == "Up")
 {
  char next = getCharAtxy(jetPlaneX, jetPlaneY - 1);
  if (next == ' ')
  {
   eraseJetPlane();
   jetPlaneY = jetPlaneY - 1;
   printJetPlane();
  }
  if (next == '*')
  {
   jetPlaneDirection = "Down";
  }
 }
 if (jetPlaneDirection == "Down")
 {
  char next = getCharAtxy(jetPlaneX, jetPlaneY + 5);
  if (next == ' ')
  {
   eraseJetPlane();
   jetPlaneY = jetPlaneY + 1;
   printJetPlane();
  }
  if (next == '*')
  {
   jetPlaneDirection = "Up";
  }
 }
}
```

```
void printJetPlane()
{
 // Changes Colour of enemy to Red
 HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
 SetConsoleTextAttribute(hConsole, 4);
 char backslash = 92;
 char jetPlane[5][6] = {{' ', ' ', ' ', ' ', '/', '|'}, {' ', '/', backslash, '/', ' ', '|'}, {'<', '[', ' ', ' ', ' ', '|'}, {' ', backslash, '/', backslash, ' ', '|'}, {' ', ' ', ' ', ' ', backslash, '|'}};

 for (int i = 0; i < 5; i++)
 {
  gotoxy(jetPlaneX, jetPlaneY + i);
  for (int j = 0; j < 6; j++)
  {
   cout << jetPlane[i][j];
  }
 }
}

void eraseJetPlane()
{
 // Erases Jet Plane
 int a = 6;
 gotoxy(jetPlaneX, jetPlaneY);
 for (int i = 0; i < a; i++)
 {
  cout << " ";
 }
gotoxy(jetPlaneX, jetPlaneY + 1);
 for (int i = 0; i < a; i++)
 {
  cout << " ";
 }
 gotoxy(jetPlaneX, jetPlaneY + 2);

 for (int i = 0; i < a; i++)
 {
  cout << " ";
 }
```

```
gotoxy(jetPlaneX, jetPlaneY + 3);

 for (int i = 0; i < a; i++)
 {
  cout << " ";
 }
 gotoxy(jetPlaneX, jetPlaneY + 4);

 for (int i = 0; i < a; i++)
 {
  cout << " ";
 }
}

void bulletCollisionWithJetPlane()
{
 // Detects Collision and if true reduces Health
 for (int x = 0; x < bulletCount; x++)
 {
  if (isBulletActive[x] == true)
  {
   if (bulletX[x] + 1 == jetPlaneX && (bulletY[x] ==
jetPlaneY + 2))
   {
    addScore();
    jetPlaneHealth--;
   }
   else if ((bulletX[x] == jetPlaneX) && (bulletY[x]
== jetPlaneY + 1 || bulletY[x] == jetPlaneY + 3))
   {
    addScore();
    jetPlaneHealth--;
   }
   else if ((bulletX[x] - 3 == jetPlaneX) &&
(bulletY[x] == jetPlaneY || bulletY[x] == jetPlaneY
+ 4))
   {
    addScore();
    jetPlaneHealth--;
   }
  }}}
```

```
void printJetPlaneHealth()
{
 char heart = 3;
 // Prints Jet Plane Health at left side of maze
 gotoxy(120, 3);
 if (jetPlaneHealth <= 15 && jetPlaneHealth > 12)
 {
  cout << "Jet Plane: " << heart << heart << heart
<< heart << heart;
 }
 gotoxy(120, 3);
 if (jetPlaneHealth <= 12 && jetPlaneHealth > 9)
 {
  cout << "Jet Plane: " << heart << heart << heart
<< heart << " ";
 }
 gotoxy(120, 3);
 if (jetPlaneHealth <= 9 && jetPlaneHealth > 6)
 {
  cout << "Jet Plane: " << heart << heart << heart
<< " "  << " ";
 }
 gotoxy(120, 3);
 if (jetPlaneHealth <= 6 && jetPlaneHealth > 3)
 {
  cout << "Jet Plane: " << heart << heart << " "
<< " "  << " ";
 }
 gotoxy(120, 3);
 if (jetPlaneHealth <= 3 && jetPlaneHealth > 0)
 {
  cout << "Jet Plane: " << heart << " " << " "  << " "
     << " ";
}
if (jetPlaneHealth <= 0)
 {
  cout << "               "  << " "  << " "  << " "  << " "
     << " ";
  gotoxy(jetPlaneX, jetPlaneY);
  eraseJetPlane();
 }}
void printInfantry(){
```

```
// Changes Infantry Health to Red
 HANDLE hConsole =
GetStdHandle(STD_OUTPUT_HANDLE);
 SetConsoleTextAttribute(hConsole, 4);
 // Infantry Character
 char backslash = 92;
 char box = 219;
 char fire = 170;
 char infantry[3][11] = {{' ', ' ', ' ', ' ', ' ', '(', '-', ' ', '-',
')', ' '},
                  {'<', '-', '-', fire, backslash, box, box,
box, box, box, '/'},
                  {' ', ' ', ' ', ' ', ' ', '|', ' ', ' ', ' ', '|', ' '}};

 for (int i = 0; i < 3; i++)
 {
  gotoxy(infantryX, infantryY + i);
  for (int j = 0; j < 11; j++)
  {
   cout << infantry[i][j];
  }
 }
}
void eraseInfantry()
{
 // Erasing Functionality for infantry
 gotoxy(infantryX, infantryY);
 for (int x = 0; x < 11; x++)
 {
  cout << " ";
 }
 gotoxy(infantryX, infantryY + 1);

 for (int x = 0; x < 11; x++)
 {
  cout << " ";
 }
 gotoxy(infantryX, infantryY + 2);

 for (int x = 0; x < 11; x++)
 {
  cout << " ";}}
```

```
void bulletCollisionWithInfantry()
{
 // Detects Collision of bullet with infantry and if
true reduces health
 for (int x = 0; x < bulletCount; x++)
 {
  if (isBulletActive[x] == true)
  {
   if ((bulletX[x] == infantryX + 4) && (infantryY ==
bulletY[x]) || (bulletX[x] == infantryX + 5) &&
(infantryY == bulletY[x]) || (bulletX[x] == infantryX
+ 6) && (infantryY == bulletY[x]) || (bulletX[x] ==
infantryX + 7) && (infantryY == bulletY[x]) ||
(bulletX[x] == infantryX + 8) && (infantryY ==
bulletY[x]) || (bulletX[x] == infantryX + 9) &&
(infantryY == bulletY[x]))
   {
    addScore();
    infantryHealth--;
   }

   else if ((bulletX[x] == infantryX - 2) && (infantryY
+ 1 == bulletY[x]) || (bulletX[x] == infantryX) &&
(infantryY + 1 == bulletY[x]) || (bulletX[x] ==
infantryX + 1) && (infantryY + 1 == bulletY[x]) ||
(bulletX[x] == infantryX + 2) && (infantryY + 1 ==
bulletY[x]) || (bulletX[x] == infantryX + 3) &&
(infantryY + 1 == bulletY[x]) || (bulletX[x] ==
infantryX + 4) && (infantryY + 1 == bulletY[x]) ||
(bulletX[x] == infantryX + 5) && (infantryY + 1 ==
bulletY[x]) || (bulletX[x] == infantryX + 6) &&
(infantryY + 1 == bulletY[x]) || (bulletX[x] ==
infantryX + 7) && (infantryY + 1 == bulletY[x]) ||
(bulletX[x] == infantryX + 8) && (infantryY + 1 ==
bulletY[x]) || (bulletX[x] == infantryX + 9) &&
(infantryY + 1 == bulletY[x]))
{
    addScore();
    infantryHealth--;
   }
}
}
}
```

```
void moveTank()
{
 // Tank Moving Functionality
 if (tankDirection == "Up")
 {
  char next = getCharAtxy(tankX, tankY - 1);
  if (next == ' ')
  {
   eraseTank();
   tankY = tankY - 1;
   printTank();
  }
  if (next == '*')
  {
   tankDirection = "Down";
  }
 }

 if (tankDirection == "Down")
 {
  char next = getCharAtxy(tankX, tankY + 3);
  if (next == ' ')
  {
   eraseTank();
   tankY = tankY + 1;
   printTank();
  }
  if (next == '*')
  {
   tankDirection = "Up";
  }
 }
}
void printInfantryHealth()
{
 // Prints Tank Heealth at the left side of maze
 char heart = 3;
 gotoxy(120, 3);
 if (infantryHealth == 5)
 {
  cout << "Infantry: " << heart << heart << heart << heart << heart;
 }
 gotoxy(120, 3);
 if (infantryHealth == 4)
 {
  cout << "Infantry: " << heart << heart << heart << heart << " ";
 }
 gotoxy(120, 3);
 if (infantryHealth == 3)
 {
  cout << "Infantry: " << heart << heart << heart << " " << " ";
 }
 gotoxy(120, 3);
 if (infantryHealth == 2)
 {
 cout << "Infantry: " << heart << heart << " " << " " << " ";
 }
 gotoxy(120, 3);
 if (infantryHealth == 1)
 {
  cout << "Infantry: " << heart << " " << " " << " " << " ";
 }
 gotoxy(120, 3);
 if (infantryHealth <= 0)
 {
  cout << "          " << " " << " " << " " << " " << " " << " ";
  gotoxy(infantryX, infantryY);
  eraseInfantry();
 }
}
void moveBalaRandom()
{
 int randomValue = rand() % 4;
 if (randomValue == 0) // 0 for moving Left
 {
  char next = getCharAtxy(balaX - 1, balaY);
  if (next == ' ' || next == '.')
  {
   eraseBala();
   balaX = balaX - 1;
```

```
  gotoxy(balaX, balaY);
  printBala();
 }
}
if (randomValue == 1) // 1 for moving Right
{
 char next = getCharAtxy(balaX + 9, balaY);
 if (next == ' ' || next == '.')
 {
  gotoxy(balaX, balaY);
  eraseBala();
  balaX = balaX + 1;
  gotoxy(balaX, balaY);
  printBala();
 }
}
if (randomValue == 2) // 2 for moving Up
{
 char next = getCharAtxy(balaX, balaY - 1);
 if (next == ' ' || next == '.')
 {
  gotoxy(balaX, balaY);
  eraseBala();
  balaY = balaY - 1;
  gotoxy(balaX, balaY);
  printBala();
 }
}
if (randomValue == 3) // 3 for moving Down
{
 char next = getCharAtxy(balaX, balaY + 2);
 if (next == ' ' || next == '.')
 {
  gotoxy(balaX, balaY);
  eraseBala();
  balaY = balaY + 1;
  gotoxy(balaX, balaY);
  printBala();
 }
}
}
```

```
void moveBalaSmart()
{
 // Distance Formula to check distance of Bala to
Rambo
 float left = distance(balaX - 1, balaY);
 float right = distance(balaX + 1, balaY);
 float up = distance(balaX, balaY - 1);
 float down = distance(balaX, balaY + 1);
 // The Player will move to the shortest distance
 if (left <= down && left <= right && left <= up) //
Move Left if true
 {
  char next = getCharAtxy(balaX - 1, balaY);
  if (next == ' ' || next == '.')
  {
   gotoxy(balaX, balaY);
   eraseBala();
   balaX = balaX - 1;
   gotoxy(balaX, balaY);
   printBala();
  }
 }
 else if (right <= down && right <= left && right <=
up) // Move Right if true
 {
  char next = getCharAtxy(balaX + 1, balaY);
  if (next == ' ' || next == '.')
  {
   gotoxy(balaX, balaY);
   eraseBala();
   balaX = balaX + 1;
   gotoxy(balaX, balaY);
   printBala();
  }
 }
 else if (down <= left && down <= right && down
<= up) // Move Down if true
 {
```

```
  char next = getCharAtxy(balaX, balaY + 1);
 if (next == ' ' || next == '.')
 {
  gotoxy(balaX, balaY);
  eraseBala();
  balaY = balaY + 1;
  gotoxy(balaX, balaY);
  printBala();
 }
}
 else // Move Up if true
 {
  char next = getCharAtxy(balaX, balaY - 1);
  if (next == ' ' || next == '.')
  {
   gotoxy(balaX, balaY);
   eraseBala();
   balaY = balaY - 1;
   gotoxy(balaX, balaY);
   printBala();
  }
 }
}

// Distance Formula
float distance(int ramboX, int ramboY)
{
 return sqrt(pow(ramboX - ramboX, 2) +
pow(ramboY - ramboY, 2));
}
void addScore()
{
 score++;
}
void printScore()
{
 // Changes Text Colour to Green
 HANDLE hConsole =
GetStdHandle(STD_OUTPUT_HANDLE);
 SetConsoleTextAttribute(hConsole, 2);
 gotoxy(120, 2);
 cout << "Score: " << score;}
```

```
 void printBala()
{
 // Prints Bala Characters
 HANDLE hConsole =
GetStdHandle(STD_OUTPUT_HANDLE);
 SetConsoleTextAttribute(hConsole, 4);
 // Bala Characters
 char backslash = 92;
 char bala[4][9] = {
    {'/', '=', '=', '=', '=', '=', '=', '=', ']'},
    {backslash, '=', '=', '=', '=', '=', '=', '=', '|'},
    {' ', ' ', ' ', ' ', ' ', '<', ' ', '|', '|'},
    {' ', ' ', ' ', ' ', ' ', ' ', ' ', '|', '|'}};
 for (int i = 0; i < 4; i++)
 {
  gotoxy(balaX, balaY + i); // Goes to next y-
coordinate to print next row
  for (int j = 0; j < 9; j++)
  {
   cout << bala[i][j];
  } }}
void eraseBala()
{
 int a = 9;
 // Erases Bala
 gotoxy(balaX, balaY);
 for (int i = 0; i < a; i++)
 {
  cout << " ";
 }
gotoxy(balaX, balaY + 1);
 for (int i = 0; i < a; i++)
 {
  cout << " ";
 }
 gotoxy(balaX, balaY + 2);
 for (int i = 0; i < a; i++)
 {
  cout << " ";
 }
 gotoxy(balaX, balaY + 3);
```

```
 gotoxy(balaX, balaY + 3);
 for (int i = 0; i < a; i++)
 {
  cout << " ";
 }
}

void bulletCollisionWithBala()
{
 // Detects Collision And Reduces Health of Bala
 for (int x = 0; x < bulletCount; x++)
 {
  if (isBulletActive[x] == true)
  {
   if (bulletX[x] + 1 == balaX && (bulletY[x] ==
balaY || bulletY[x] == balaY + 1))
   {
    addScore();
    balaHealth--;
   }
   else if (balaX - 1 == bulletX[x] && balaY + 1 ==
bulletY[x])
   {
    addScore();
    balaHealth--;
   }
   else if (bulletX[x] - 4 == balaX && (bulletY[x] ==
balaY + 2))
   {
    addScore();
    balaHealth--;
   }
else if (bulletX[x] - 6 == balaX && (bulletY[x] ==
balaY + 3))
   {
    addScore();
    balaHealth--;
   }
  }
 }
}
```

```
void printBalaHealth()
{
 char heart = 3;
 // Prints Bala Health at Left Side of Maze
 gotoxy(120, 3);
 if (balaHealth <= 20 && balaHealth > 16)
 {
  cout << "Bala: " << heart << heart << heart <<
heart << heart;
 }
 gotoxy(120, 3);
 if (balaHealth <= 16 && balaHealth > 12)
 {
  cout << "Bala: " << heart << heart << heart <<
heart << " ";
 }
 gotoxy(120, 3);
 if (balaHealth <= 12 && balaHealth > 8)
 {
  cout << "Bala: " << heart << heart << heart << " "
      << " ";
 }
 gotoxy(120, 3);
 if (balaHealth <= 8 && balaHealth > 4)
 {
  cout << "Bala: " << heart << heart << " " << " "
      << " ";
 }
 gotoxy(120, 3);
 if (balaHealth <= 4 && balaHealth > 0)
 {
  cout << "Bala: " << heart << " " << " " << " " << "
";}
 gotoxy(120, 3);
 if (balaHealth <= 0)
 {
  cout << "      "<< " " << " " << " "  << " " << " ";
  gotoxy(balaX, balaY);
  eraseBala();
 }
}
```

```
void generateEnemyBullet(int x, int y)
{
 // Generates Bullet for Enemies
 enemyBulletX[enemyBulletCount] = x - 1;
 enemyBulletY[enemyBulletCount] = y;
 isEnemyBulletActive[enemyBulletCount] = true;
 gotoxy(x - 1, y);
 cout << "~";
 enemyBulletCount++;
}

void moveEnemyBullet()
{
 // Moves Active Bullets
 for (int i = 0; i < enemyBulletCount; i++)
 {
  if (isEnemyBulletActive[i] == true)
  {
   char next = getCharAtxy(enemyBulletX[i] - 1,
enemyBulletY[i]);

   if (next == ' ')
   {
    eraseEnemyBullet(enemyBulletX[i],
enemyBulletY[i]);
    enemyBulletX[i] = enemyBulletX[i] - 1;
    printEnemyBullet(enemyBulletX[i],
enemyBulletY[i]);
   }
   else if (next != ' ')
   {
    eraseEnemyBullet(enemyBulletX[i],
enemyBulletY[i]);
    makeEnemyBulletInactive(i);
   }
  }
 }
}
```

```
void makeEnemyBulletInactive(int idx)
{
 // Makes Enemy Bullet Inactive
 isEnemyBulletActive[idx] = false;
}

void eraseEnemyBullet(int x, int y)
{
 // Erases Bullet
 gotoxy(x, y);
 cout << " ";
}

void printHealthPill()
{
 // Prints Health Pill
 char heart = 3;
 gotoxy(70, 20);
 cout << heart;
}

void removeHealthPill()
{
 // Removes Health Pill
 gotoxy(70, 20);
 cout << " ";
}

void healthPill()
{
 // Health Pill Functionality
 printHealthPill();
 if (((ramboX + 8 == pillX) && (ramboY == pillY)) ||
((ramboX + 8 == pillX) && (ramboY + 1 == pillY))
|| ((ramboX + 4 == pillX) && (ramboY + 2 ==
pillY)) || ((ramboX + 2 == pillX) && (ramboY + 3
== pillY)))
 {
  towerHealth = towerHealth + 10;
  removeHealthPill();
 }
```

```
else if (((ramboX == pillX) || (ramboX + 1 == pillX)
|| (ramboX + 2 == pillX) || (ramboX + 3 == pillX)
|| (ramboX + 4 == pillX) || (ramboX + 5 == pillX)
|| (ramboX + 6 == pillX) || (ramboX + 7 == pillX))
&& (ramboY == pillY))
 {
  towerHealth = towerHealth + 10;
  removeHealthPill();
 }
 else if (((ramboX == pillX) || (ramboX + 1 == pillX)
|| (ramboX + 2 == pillX) || (ramboX + 3 == pillX)
|| (ramboX + 4 == pillX) || (ramboX + 5 == pillX)
|| (ramboX + 6 == pillX) || (ramboX + 7 == pillX))
&& (ramboY + 3 == pillY))
 {
  towerHealth = towerHealth + 10;
  removeHealthPill();
 }
}

void bulletCollisionWithTower()
{
 // Detects Bullet Collision With Tower and
reduces Health
 for (int x = 0; x < enemyBulletCount; x++)
 {
  if (isEnemyBulletActive[x] == true)
  {
   char next = getCharAtxy(enemyBulletX[x] - 1,
enemyBulletY[x]);
   if (next == '!')
   {
    towerHealth = towerHealth -
enemyFireDamage;
   }
  }
 }
}
bool playerCollisionWithEnemyRight()
{
 bool collision = false;
 char heart = 3;
```

```
// Takes Characters around rambo right
 char isEnemy1 = getCharAtxy(ramboX + 9,
ramboY);
 char isEnemy2 = getCharAtxy(ramboX + 9,
ramboY + 1);
 char isEnemy3 = getCharAtxy(ramboX + 4,
ramboY + 2);
 char isEnemy4 = getCharAtxy(ramboX + 2,
ramboY + 3);
 if ((isEnemy1 == '~') || (isEnemy2 == '~') ||
(isEnemy3 == '~') || (isEnemy4 == '~'))
 {
  return false;
 }
 if ((isEnemy1 == heart) || (isEnemy2 == heart) ||
(isEnemy3 == heart) || (isEnemy4 == heart))
 {
  return false;
 }
 if ((isEnemy1 != ' ') || (isEnemy2 != ' ') ||
(isEnemy3 != ' ') || (isEnemy4 != ' '))
 {
  collision = true;
 }
 if ((isEnemy1 == '!') || (isEnemy2 == '!') ||
(isEnemy3 == '!') || (isEnemy4 == '!'))
 {
  collision = false;
 }
 if ((isEnemy1 == '*') || (isEnemy2 == '*') ||
(isEnemy3 == '*') || (isEnemy4 == '*'))
 {
  collision = false;
 }

 return collision;
}
bool playerCollisionWithEnemyLeft()
{
 bool collision = false;
 char heart = 3;
```

```
 // Takes Characters around rambo left
 char isEnemy1 = getCharAtxy(ramboX - 1, ramboY);
 char isEnemy2 = getCharAtxy(ramboX - 1, ramboY + 1);
 char isEnemy3 = getCharAtxy(ramboX - 1, ramboY + 2);
 char isEnemy4 = getCharAtxy(ramboX - 1, ramboY + 3);
 if ((isEnemy1 == '~') || (isEnemy2 == '~') || (isEnemy3 == '~') || (isEnemy4 == '~'))
 {
  return false;
 }
 if ((isEnemy1 == heart) || (isEnemy2 == heart) || (isEnemy3 == heart) || (isEnemy4 == heart))
 {
  return false;
 }
 if ((isEnemy1 != ' ') || (isEnemy2 != ' ') || (isEnemy3 != ' ') || (isEnemy4 != ' '))
 {
  collision = true;
 }
 if ((isEnemy1 == '!') || (isEnemy2 == '!') || (isEnemy3 == '!') || (isEnemy4 == '!'))
 {
  collision = false;
 }
 if ((isEnemy1 == '*') || (isEnemy2 == '*') || (isEnemy3 == '*') || (isEnemy4 == '*'))
 {
  collision = false;
 }

 return collision;
}
bool playerCollisionWithEnemyUp()
{
 // Takes Characters around rambo Up
 bool collision = false;
 char heart = 3;
 char isEnemy1 = getCharAtxy(ramboX, ramboY + 4);
 char isEnemy2 = getCharAtxy(ramboX + 1, ramboY + 4);
 char isEnemy3 = getCharAtxy(ramboX + 3, ramboY + 3);
 char isEnemy4 = getCharAtxy(ramboX + 4, ramboY + 2);
 char isEnemy5 = getCharAtxy(ramboX + 5, ramboY + 2);
 char isEnemy6 = getCharAtxy(ramboX + 6, ramboY + 2);
 char isEnemy7 = getCharAtxy(ramboX + 4, ramboY + 2);
 char isEnemy8 = getCharAtxy(ramboX + 4, ramboY + 2);

 if ((isEnemy1 == '~') || (isEnemy2 == '~') || (isEnemy3 == '~') || (isEnemy4 == '~') || (isEnemy5 == '~') || (isEnemy6 == '~') || (isEnemy7 == '~') || (isEnemy8 == '~'))
 {
  return false;
 }
 if ((isEnemy1 == heart) || (isEnemy2 == heart) || (isEnemy3 == heart) || (isEnemy4 == heart) || (isEnemy5 == heart) || (isEnemy6 == heart) || (isEnemy7 == heart) || (isEnemy8 == heart))
 {
  return false;
 }
 if ((isEnemy1 != ' ') || (isEnemy2 != ' ') || (isEnemy3 != ' ') || (isEnemy4 != ' ') || (isEnemy5 != ' ') || (isEnemy6 != ' ') || (isEnemy7 != ' ') || (isEnemy8 != ' '))
 {
  collision = true;
 }
 if ((isEnemy1 == '!') || (isEnemy2 == '!') || (isEnemy3 == '!') || (isEnemy4 == '!') || (isEnemy5 == '!') || (isEnemy6 == '!') || (isEnemy7 == '!') || (isEnemy8 == '!'))
```

```
{
  collision = false;
 }
 if ((isEnemy1 == '*') || (isEnemy2 == '*') ||
(isEnemy3 == '*') || (isEnemy4 == '*') ||
(isEnemy5 == '*') || (isEnemy6 == '*') ||
(isEnemy7 == '*') || (isEnemy8 == '*'))
 {
  collision = false;
 }
 return collision;
}
bool playerCollisionWithEnemyDown()
{
 bool collision = false;
 char heart = 3;
 // Takes Characters around rambo Down
 char isEnemy1 = getCharAtxy(ramboX, ramboY -
1);
 char isEnemy2 = getCharAtxy(ramboX + 1,
ramboY - 1);
 char isEnemy3 = getCharAtxy(ramboX + 2,
ramboY - 1);
 char isEnemy4 = getCharAtxy(ramboX + 3,
ramboY - 1);
 char isEnemy5 = getCharAtxy(ramboX + 4,
ramboY - 1);
 char isEnemy6 = getCharAtxy(ramboX + 5,
ramboY - 1);
 char isEnemy7 = getCharAtxy(ramboX + 6,
ramboY - 1);
 char isEnemy8 = getCharAtxy(ramboX + 7,
ramboY - 1);
 char isEnemy9 = getCharAtxy(ramboX + 8,
ramboY - 1);

 if ((isEnemy1 == '~') || (isEnemy2 == '~') ||
(isEnemy3 == '~') || (isEnemy4 == '~') ||
(isEnemy5 == '~') || (isEnemy6 == '~') ||
(isEnemy7 == '~') || (isEnemy8 == '~') ||
(isEnemy9 == '~'))
 {
  return false;
 }
 if ((isEnemy1 == heart) || (isEnemy2 == heart) ||
(isEnemy3 == heart) || (isEnemy4 == heart) ||
(isEnemy5 == heart) || (isEnemy6 == heart) ||
(isEnemy7 == heart) || (isEnemy8 == heart) ||
(isEnemy9 == heart))
 {
  return false;
 }
 if ((isEnemy1 != ' ') || (isEnemy2 != ' ') ||
(isEnemy3 != ' ') || (isEnemy4 != ' ') || (isEnemy5
!= ' ') || (isEnemy6 != ' ') || (isEnemy7 != ' ') ||
(isEnemy8 != ' ') || (isEnemy9 != ' '))
 {
  collision = true;
 }
 if ((isEnemy1 == '!') || (isEnemy2 == '!') ||
(isEnemy3 == '!') || (isEnemy4 == '!') || (isEnemy5
== '!') || (isEnemy6 == '!') || (isEnemy7 == '!') ||
(isEnemy8 == '!') || (isEnemy9 == '!'))
 {
  collision = false;
 }
 if ((isEnemy1 == '*') || (isEnemy2 == '*') ||
(isEnemy3 == '*') || (isEnemy4 == '*') ||
(isEnemy5 == '*') || (isEnemy6 == '*') ||
(isEnemy7 == '*') || (isEnemy8 == '*') ||
(isEnemy9 == '*'))
 {
  collision = false;
 }

 return collision;
}
void printRamboHealth()
{
 // Changes Text Colour to Blue
 char heart = 3;
 HANDLE hConsole =
GetStdHandle(STD_OUTPUT_HANDLE);
 SetConsoleTextAttribute(hConsole, 1);
```

```
 gotoxy(120, 4);
 if (towerHealth <= 300 && towerHealth > 240)
 {
  cout << "Tower: " << heart << heart << heart << heart << heart;
 }
 gotoxy(120, 4);
 if (towerHealth <= 240 && towerHealth > 180)
 {
  cout << "Tower: " << heart << heart << heart << heart << " ";
 }
 gotoxy(120, 4);
 if (towerHealth <= 180 && towerHealth > 120)
 {
  cout << "Tower: " << heart << heart << heart << "  "  << " ";
 }
 gotoxy(120, 4);
 if (towerHealth <= 120 && towerHealth > 60)
 {
  cout << "Tower: " << heart << heart << " " << " "
      << " ";
 }
 gotoxy(120, 4);
 if (towerHealth <= 60 && towerHealth > 0)
 {
  cout << "Tower: " << heart << " " << " "<< " "
      << " ";
 }
 gotoxy(120, 4);
 if (towerHealth <= 0)
 {
  cout << "      "  << " "  << " "  << " "  << " " << " ";
 }
}
void gameOver()
{
 // Game Over is printed
 system("cls");
 gotoxy(40, 10);
 cout << "     *********    ***    **   **
*********     ";
 gotoxy(40, 11);
 cout << "     *        ** **   * * * * *        ";
 gotoxy(40, 12);
 cout << "     *   ****   *******   * ** *
******       ";
 gotoxy(40, 13);
 cout << "     *     *  **   **  *   * *        ";
 gotoxy(40, 14);
 cout << "     ********* **     ** *    *
*********     ";
 gotoxy(40, 17);
 cout << "     ********* **     ** *********
*******      ";
 gotoxy(40, 18);
 cout << "     *     *  **   **  *     *    *    ";
 gotoxy(40, 19);
 cout << "     *     *   **  **   ******
********      ";
 gotoxy(40, 20);
 cout << "     *     *   ** **   *     * **      ";
 gotoxy(40, 21);
 cout << "     *********    **    ********* *
**      ";
 gotoxy(40, 23);
 cout << "      Enter any key to exit.";
 string exit;
 getline(cin, exit);
}
void youWon()
{
 // Prints Congrats

 gotoxy(40, 10);
 cout << "     ********* *********  ***   *
*********  *******    ***    **********
********* ";
 gotoxy(40, 11);
 cout << "     *      *    *  * *  * *     *    *
** **      **    **     ";
```

```
gotoxy(40, 12);
 cout << "     *      *    *  *  *  * *  *****
*********   *******     **    ********* ";
 gotoxy(40, 13);
 cout << "     *      *    *  *   * * *    * *  **
**    **     **        ** ";
 gotoxy(40, 14);
 cout << "     ********* *********  *    ***
********* *   **  **    **    **    *********
";
 gotoxy(40, 16);
 cout << "      Enter 0 to exit.";
}
void printEnemyBullet(int x, int y)
{
 // Changes Enemy Bullet Colour to Red
 HANDLE hConsole =
GetStdHandle(STD_OUTPUT_HANDLE);
 SetConsoleTextAttribute(hConsole, 4);
 gotoxy(x, y);
 cout << "~";
}
void printRambo()
{
 // Changes Rambo Colour to Blue
 HANDLE hConsole =
GetStdHandle(STD_OUTPUT_HANDLE);
 SetConsoleTextAttribute(hConsole, 1);

 // Rambo Character
 char backslash = 92;
 char rambo[4][9] = {{'[', '=', '=', '=', '=', '=', '=', '=',
backslash}, {'|', '=', '=', '=', '=', '=', '=', '=', '/'}, {'|',
'|', ' ', '>', ' ', ' ', ' ', ' ', ' '}, {'|', '|', ' ', ' ', ' ', ' ', ' ', ' ', '
'}};
 for (int i = 0; i < 4; i++)
 {
  gotoxy(ramboX, ramboY + i);
  for (int j = 0; j < 9; j++)
  {
   cout << rambo[i][j]; } }}
```

```
void removeRambo()
{
 // Erases Rambo
 int a = 9;
 gotoxy(ramboX, ramboY);
 for (int i = 0; i < a; i++)
 {
  cout << " ";
 }

 gotoxy(ramboX, ramboY + 1);
 for (int i = 0; i < a; i++)
 {
  cout << " ";
 }

 gotoxy(ramboX, ramboY + 2);
 for (int i = 0; i < a; i++)
 {
  cout << " ";
 }

 gotoxy(ramboX, ramboY + 3);
 for (int i = 0; i < a; i++)
 {
  cout << " ";
 }}
void moveRamboUp()
{
 // Rambo Up Moving Functionality
 char heart = 3;
 char nextLocation = getCharAtxy(ramboX,
ramboY - 1);
 if (nextLocation == ' ' || (nextLocation == heart))
 {
  gotoxy(ramboX, ramboY);
  removeRambo();
  ramboY = ramboY - 1;
  printRambo(); }
}
```

```
void moveRamboDown()
{
 // Rambo Down Moving Functionality
 char heart = 3;
 char nextLocation = getCharAtxy(ramboX,
ramboY + 4);
 if (nextLocation == ' ' || (nextLocation == heart))
 {
  removeRambo();
  ramboY = ramboY + 1;
  printRambo();
 }
}

void moveRamboLeft()
{
 // Rambo Left Moving Functionality
 char heart = 3;
 char nextLocation = getCharAtxy(ramboX - 1,
ramboY);
 if (nextLocation == ' ' || (nextLocation == heart))
 {
  removeRambo();
  ramboX = ramboX - 1;
  printRambo();
 }
}

void moveRamboRight()
{
 // Rambo Right Moving Functionality
 char heart = 3;
 char nextLocation = getCharAtxy(ramboX + 9,
ramboY);
 if ((nextLocation == ' ') || (nextLocation ==
heart))
 {
  removeRambo();
  ramboX = ramboX + 1;
  printRambo();
 }
}
```

```
 void generateBullet()
{
 // Generates Player's Bullet
 bulletX[bulletCount] = ramboX + 10;
 bulletY[bulletCount] = ramboY;
 isBulletActive[bulletCount] = true;
 gotoxy(ramboX + 10, ramboY);
 cout << ".";
 bulletCount++;
}
void moveInfantry()
{
 // Infantry Moving Functionality
 if (infantryDirection == "Right")
 {
  char next = getCharAtxy(infantryX + 14,
infantryY);
  if (next == ' ')
  {
   eraseInfantry();
   infantryX = infantryX + 1;
   printInfantry();
  }
  if (next == '*')
  {
   infantryDirection = "Left";
  }
 }
 if (infantryDirection == "Left")
 {
  char next = getCharAtxy(infantryX - 1, infantryY);
  if (next == ' ')
  {
   eraseInfantry();
   infantryX = infantryX - 1;
   printInfantry();
  }
  if (next == '!')
  {
   infantryDirection = "Right";
  }
 }}
```

```
void moveBullet()
{
 // Moving functionality for Player's Bullet
 for (int i = 0; i < bulletCount; i++)
 {
  if (isBulletActive[i] == true)
  {
   char next = getCharAtxy(bulletX[i] + 1,
bulletY[i]);
   if (next != ' ')
   {
    eraseBullet(bulletX[i], bulletY[i]);
    makeBulletInactive(i);
   }

   else
   {
    eraseBullet(bulletX[i], bulletY[i]);
    bulletX[i] = bulletX[i] + 1;
    printBullet(bulletX[i], bulletY[i]);
   }
  }
 }
}

void printBullet(int x, int y)
{
 // Changes Bullet Colour to Light Blue
 HANDLE hConsole =
GetStdHandle(STD_OUTPUT_HANDLE);
 SetConsoleTextAttribute(hConsole, 11);
 gotoxy(x, y);
 cout << ".";
}

void eraseBullet(int x, int y)
{
 // Erases Bullet
 gotoxy(x, y);
 cout << " ";
}
```

```
void makeBulletInactive(int idx)
{
 // Makes Player's Bullet Inactive
 isBulletActive[idx] = false;
}

void nextLevel()
{
 // Prints NExt Level
 system("cls");
 gotoxy(0, 5);
 cout << "             ###    ## ######## ##   ##
########       " << endl;
 cout << "             ## ##  ## ##      ## ##    ##
" << endl;
 cout << "             ## ## ## #####     ##     ##
" << endl;
 cout << "             ##  ## ## ##      ## ##    ##
" << endl;
 cout << "             ##   ### ######## ##   ##
##          " << endl;
 cout << "                                          "
<< endl;
 cout << "        ##       ######## ##     ##
######## ##       " << endl;
 cout << "        ##       ##      ##   ## ##     ##
" << endl;
 cout << "        ##       #####    ##  ##  #####
##       " << endl;
cout << "        ##       ##       ## ##   ##     ##
" << endl;
 cout << "        ########## ########    ###
######## ########## " << endl;
 cout << "         Enter any key to exit....";
 string anykey;
 getline(cin, anykey);
 system("cls");
 printMaze();
}
bool setCursor(bool visible)
{
 CONSOLE_CURSOR_INFO info;
```

info.dwSize = 100;
info.bVisible = visible;
 	
SetConsoleCursorInfo(GetStdHandle(STD_OUTPUT_HANDLE), &info);
}	