**Bounce Ball GAME**



Session: 2021 – 2025

**Submitted by:**

UMAR JAMIL 2021-CS-89

**Supervised by:**

**Dr. Maida Shahid**

Department of Computer Science

**University of Engineering and Technology**

**Lahore Pakistan**

**Short Description and Story Writing of Game:**

It is a bounce ball game in which I have ball represented by B and five type of enemies. Three enemies are spider and one is continuously firing and one is randomly moving. There are two levels of this game. If ball eats all the \* then ball will move towards the next level. Total health of the ball is 5 after collision with each hurdle life will decrease by 1 and if ball collide with character E then score will increase by 10.

**Game Characters Description:**

The game is based upon two characters:

* Ball

**Description:** It is of size character.

* Enemies

**Description:** There are 5 types of enemies. 4 spiders in which one is randomly moving. An enemy X who is continuously firing.

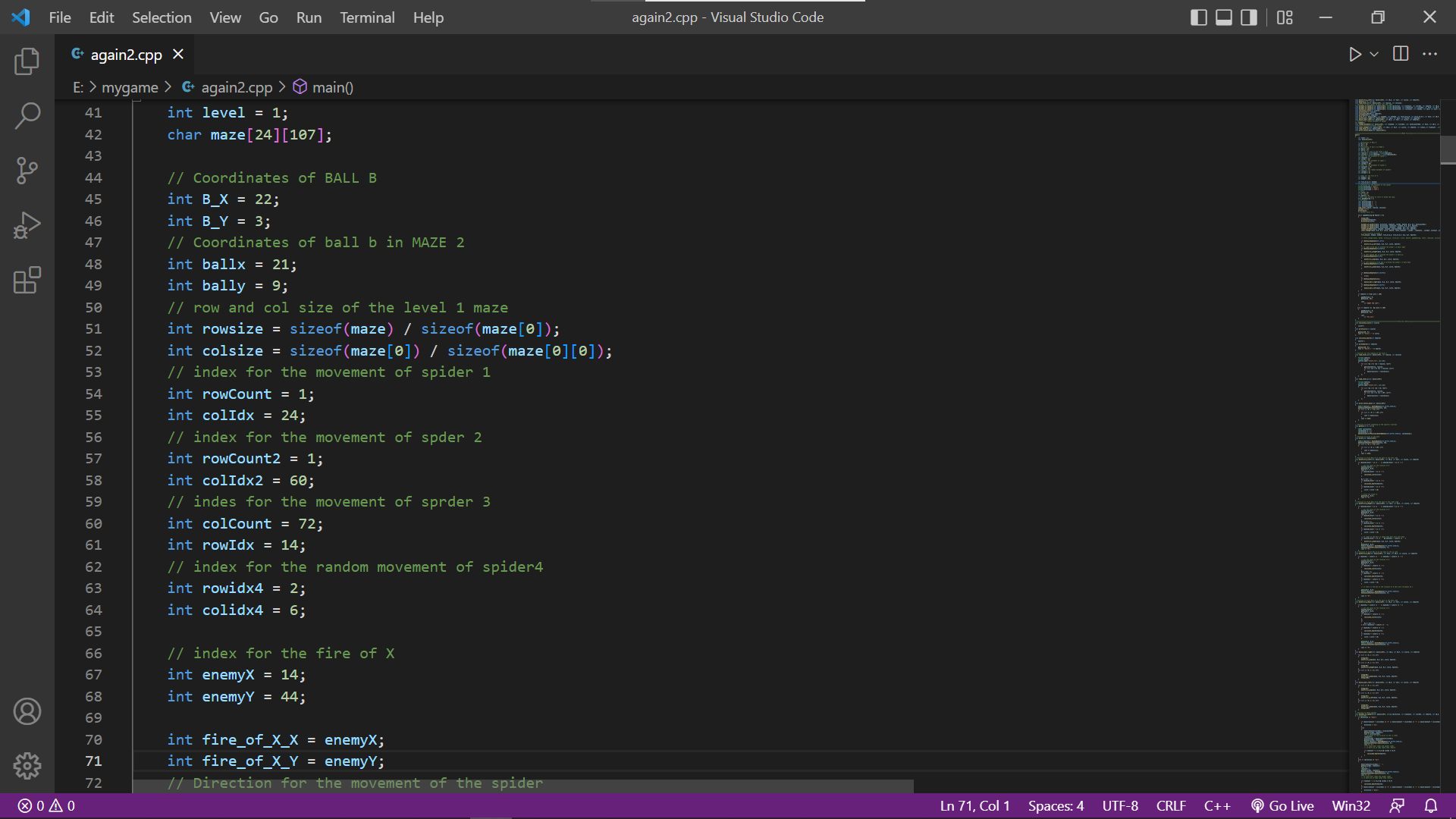
**Rules & Interactions:**

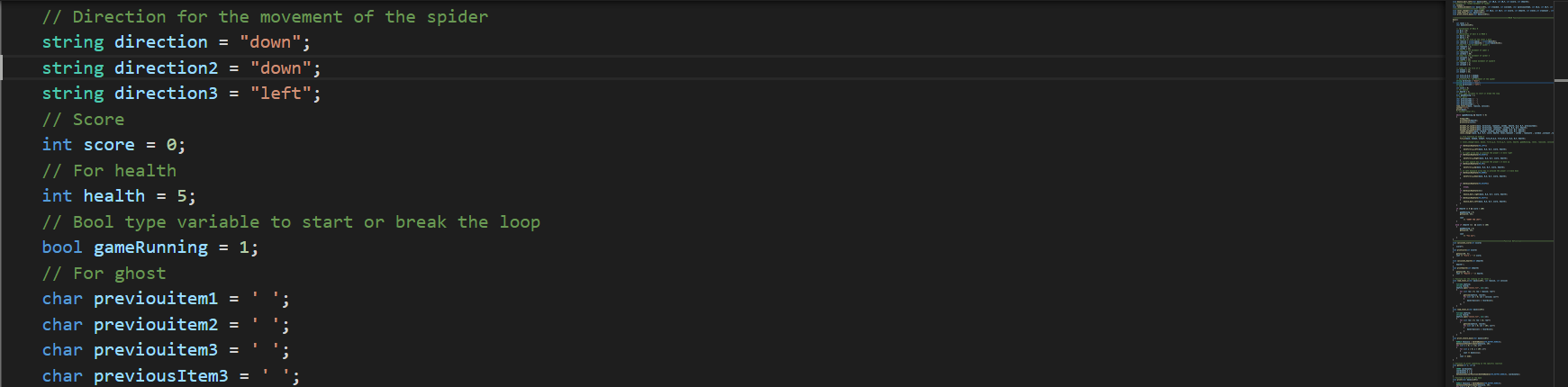
* Lives of the ball is 5.
* Score increased as ball easts \*.
* As ball collide with any spider or fire then life decreases by 1.
* Ball bounce forward with space key and backward with shift.
* Game over when lives become 0.
* Press escape key to exit the game.

**Goal of the Game:**

The goal of the game is to apply various concepts like 2D Arrays and many others too in such a way that it shows the impressive output. Also, to learn from the previous concepts etc.

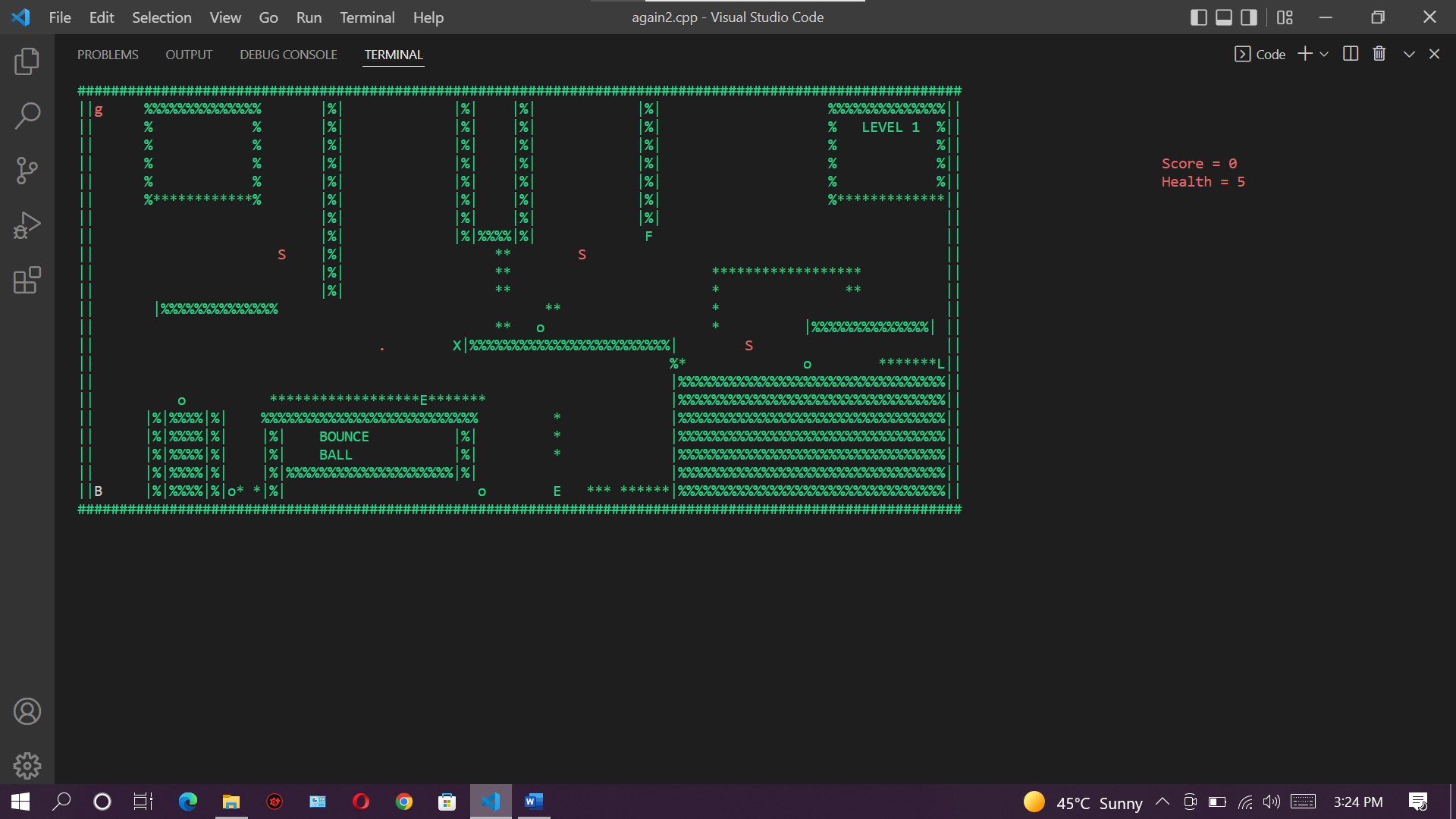
**Data structures:**





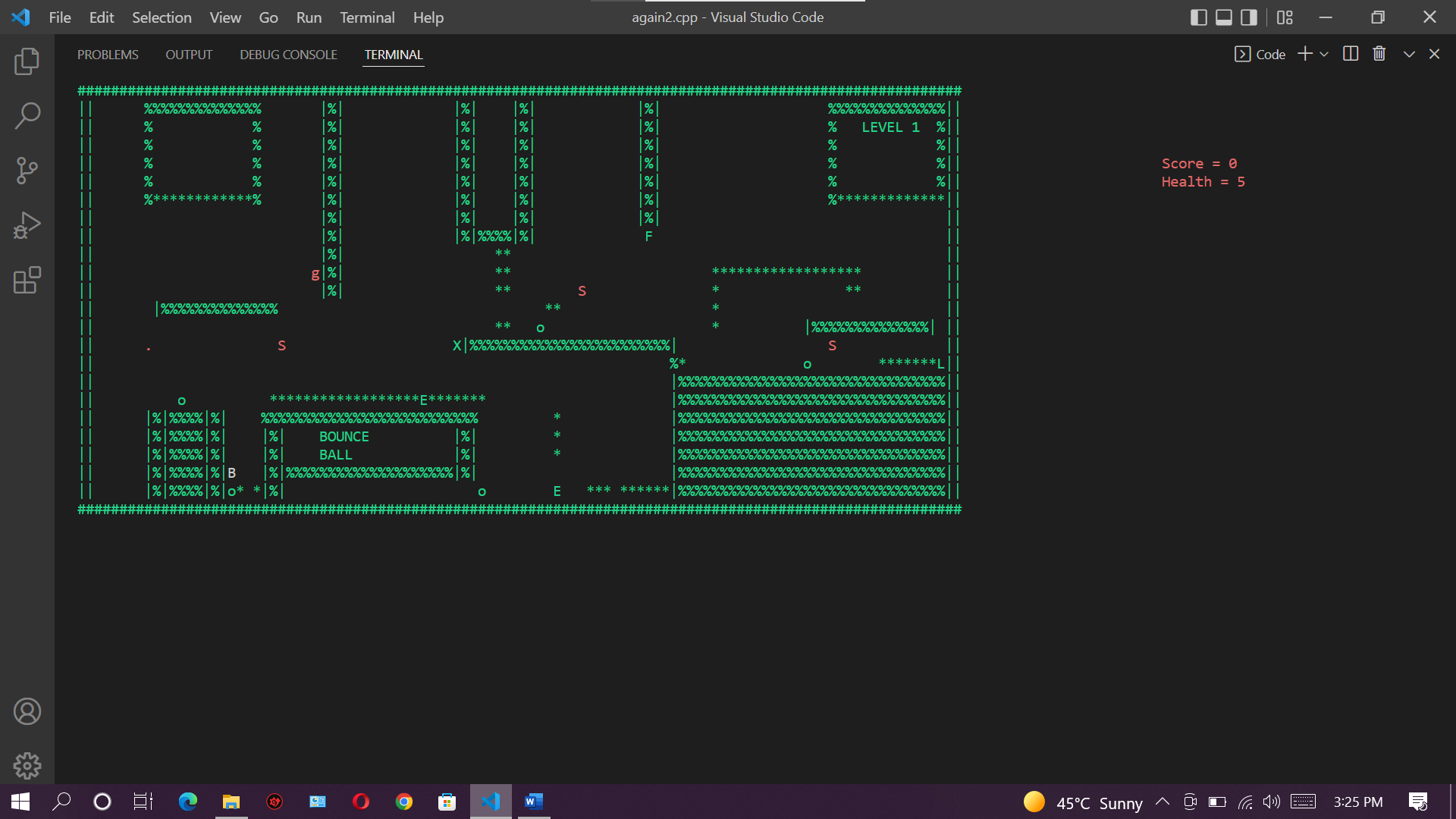
**SCREEN SHOTS OF THE GAME:**

* **MAIN SCREEN OF GAME:**

****

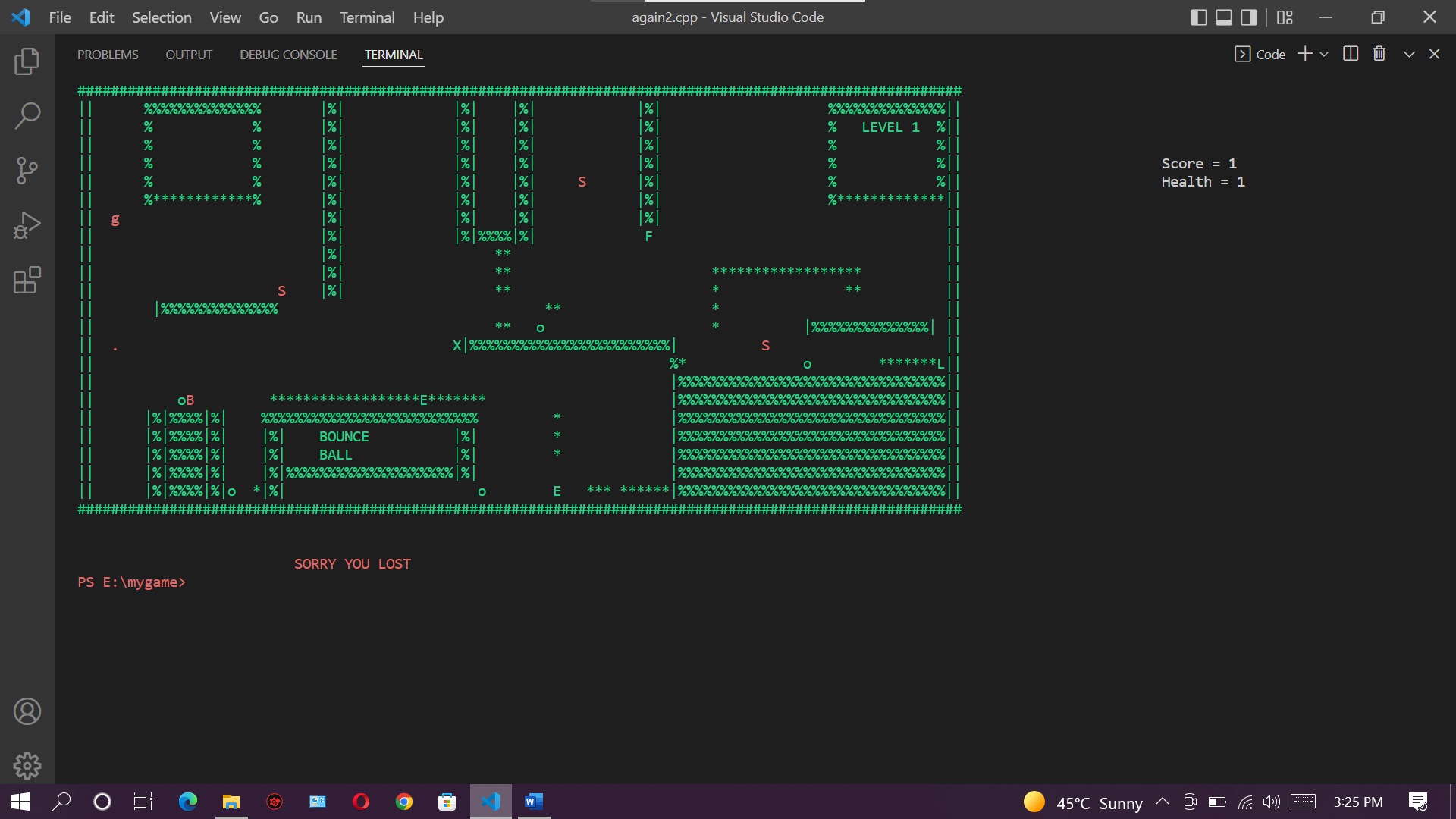
**Fig 1:** Main screen of the game

**Game play :**



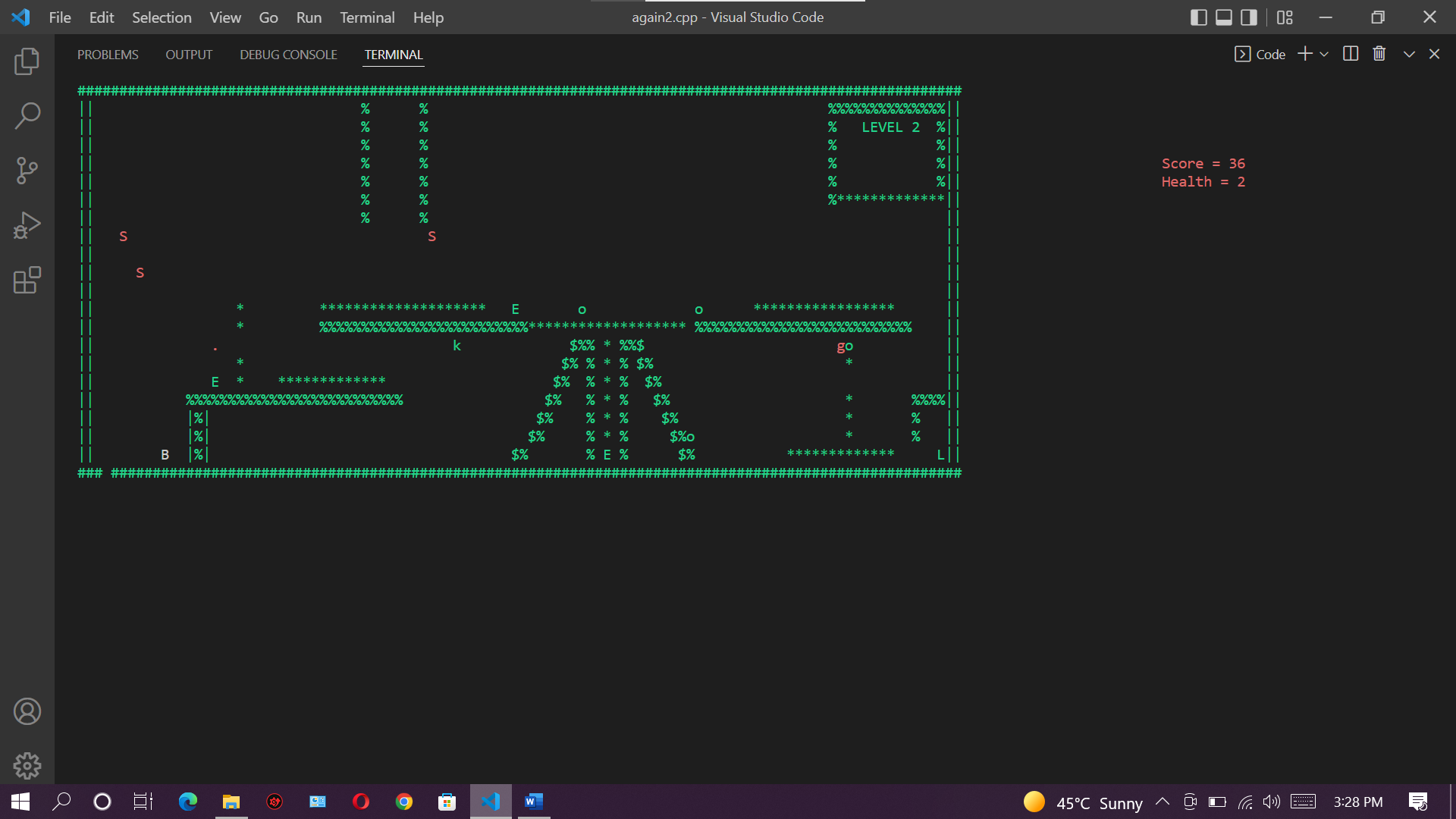
**Fig 3:** Screen shot of game play

**Game over screen:**

****

**Fig 5:** game over screen

**LEVEL 2 :**

****

**Complete Code of Game:**

**#include <iostream>**

**#include <time.h>**

**#include <conio.h>**

**#include <fstream>**

**#include <windows.h>**

**using namespace std;**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Function Prototypes\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**void header();**

**void print(char maze[][107]);**

**void print\_level2\_maze(char maze2[][107]);**

**// Ball movement function**

**void movefirst\_p\_Down(char maze[][107], int &B\_X, int &B\_Y, int &score, int &health);**

**void movefirst\_p\_Up(char maze[][107], int &B\_X, int &B\_Y, int &score, int &health);**

**void movefirst\_p\_Right(char maze[][107], int &B\_X, int &B\_Y, int &score, int &health);**

**void movefirst\_p\_Left(char maze[][107], int &B\_X, int &B\_Y, int &score, int &health);**

**void gotoxy(int x, int y);**

**void load\_level\_1(char maze[][107], int rowsize, int colsize);**

**// Function to move spider up down left and right**

**void movemet\_of\_spider1(char maze[][107], string &direction, int &rowCount, int &colIdx, int &health, int &B\_X, int &B\_Y, char previouitem1);**

**void movemet\_of\_spider2(char maze[][107], string &direction2, int &rowCount2, int &colIdx2, int &B\_X, int &B\_Y, int &health);**

**void movemet\_of\_spider3(char maze[][107], string &direction3, int &colCount, int &rowIdx, int &B\_X, int &B\_Y, int &health);**

**void calculate\_score(int &score);**

**void printscore(int &score);**

**void calculate\_health(int &health);**

**void printhealth(int &health);**

**void fire\_X(char maze[][107], int &enemyX, int &enemyY, int &fire\_of\_X\_X, int &fire\_of\_X\_Y, int &B\_X, int &B\_Y, int &health);**

**void bounce\_ball\_right(char maze[][107], int &B\_X, int &B\_Y, int &score, int &health);**

**void bounce\_ball\_left(char maze[][107], int &B\_X, int &B\_Y, int &score, int &health);**

**// Functoin for random movement of spider**

**int random();**

**void random\_movement(char maze[][107], int &rowidx4, int &colidx4, char &previousItem3, int &B\_X, int &B\_Y, int &health);**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*level changer**

**void level\_changer(char maze[][107], int &B\_X, int &B\_Y, int &score, int &health, int &level,int &rowCount , int &colIdx , int &rowCount2 , int &colIdx2 ,int &colCount ,int &rowIdx , int &rowidx4 , int &colidx4 );**

**void load\_level\_2(char maze[][107]);**

**void print\_level2\_maze(char maze[][107]);**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*MAIN Function\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**main()**

**{**

**int level = 1;**

**char maze[24][107];**

**// Coordinates of BALL B**

**int B\_X = 22;**

**int B\_Y = 3;**

**// Coordinates of ball b in MAZE 2**

**int ballx = 21;**

**int bally = 9;**

**// row and col size of the level 1 maze**

**int rowsize = sizeof(maze) / sizeof(maze[0]);**

**int colsize = sizeof(maze[0]) / sizeof(maze[0][0]);**

**// index for the movement of spider 1**

**int rowCount = 1;**

**int colIdx = 24;**

**// index for the movement of spder 2**

**int rowCount2 = 1;**

**int colIdx2 = 60;**

**// indes for the movement of sprder 3**

**int colCount = 72;**

**int rowIdx = 14;**

**// index for the random movement of spider4**

**int rowidx4 = 2;**

**int colidx4 = 6;**

**// index for the fire of X**

**int enemyX = 14;**

**int enemyY = 44;**

**int fire\_of\_X\_X = enemyX;**

**int fire\_of\_X\_Y = enemyY;**

**// Direction for the movement of the spider**

**string direction = "down";**

**string direction2 = "down";**

**string direction3 = "left";**

**// Score**

**int score = 0;**

**// For health**

**int health = 5;**

**// Bool type variable to start or break the loop**

**bool gameRunning = 1;**

**// For ghost**

**char previouitem1 = ' ';**

**char previouitem2 = ' ';**

**char previouitem3 = ' ';**

**char previousItem3 = ' ';**

**load\_level\_1(maze, rowsize, colsize);**

**system("cls");**

**print(maze);**

**// system("color 9");**

**while (gameRunning && health != 0)**

**{**

**Sleep(100);**

**printhealth(health);**

**printscore(score);**

**movemet\_of\_spider1(maze, direction, rowCount, colIdx, health, B\_X, B\_Y, previouitem1);**

**movemet\_of\_spider2(maze, direction2, rowCount2, colIdx2, B\_X, B\_Y, health);**

**movemet\_of\_spider3(maze, direction3, colCount, rowIdx, B\_X, B\_Y, health);**

**random\_movement(maze, rowidx4, colidx4, previousItem3, B\_X, B\_Y, health);**

**level\_changer(maze, B\_X, B\_Y, score, health, level,rowCount , colIdx , rowCount2 , colIdx2 ,colCount ,rowIdx , rowidx4 ,colidx4 );**

**// fire function of enemy X**

**fire\_X(maze, enemyX, enemyY, fire\_of\_X\_X, fire\_of\_X\_Y, B\_X, B\_Y, health);**

**// level\_changer(maze, maze2, first\_p\_X, first\_p\_Y, score, health, gameRunning, level, rowsize2, colsize2);**

**if (GetAsyncKeyState(VK\_LEFT))**

**{**

**movefirst\_p\_Left(maze, B\_X, B\_Y, score, health);**

**}**

**// if right arrow key is pressed the player 1 A move right**

**if (GetAsyncKeyState(VK\_RIGHT))**

**{**

**movefirst\_p\_Right(maze, B\_X, B\_Y, score, health);**

**}**

**// if left upward key is pressed the player 1 A move up**

**if (GetAsyncKeyState(VK\_UP))**

**{**

**movefirst\_p\_Up(maze, B\_X, B\_Y, score, health);**

**}**

**// if left downward arrow key is pressed the player 1 A move down**

**if (GetAsyncKeyState(VK\_DOWN))**

**{**

**movefirst\_p\_Down(maze, B\_X, B\_Y, score, health);**

**}**

**if (GetAsyncKeyState(VK\_ESCAPE))**

**{**

**break;**

**}**

**if (GetAsyncKeyState(32))**

**{**

**bounce\_ball\_right(maze, B\_X, B\_Y, score, health);**

**}**

**if (GetAsyncKeyState(VK\_SHIFT))**

**{**

**bounce\_ball\_left(maze, B\_X, B\_Y, score, health);**

**}**

**}**

**if (health == 0 && score < 180)**

**{**

**gameRunning = 0;**

**gotoxy(26, 26);**

**cout**

**<< "SORRY YOU LOST";**

**}**

**else if (health >=1 && score >= 180)**

**{**

**gameRunning = 0;**

**gotoxy(26, 26);**

**cout**

**<< "You won";**

**}**

**}**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Function Definition\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**void calculate\_score(int &score)**

**{**

**score++;**

**}**

**void printscore(int &score)**

**{**

**gotoxy(130, 4);**

**cout << "Score = " << score;**

**}**

**void calculate\_health(int &health)**

**{**

**health--;**

**}**

**void printhealth(int &health)**

**{**

**gotoxy(130, 5);**

**cout << "Health = " << health;**

**}**

**// Function for the loading of the level 1**

**void load\_level\_1(char maze[][107], int rowsize, int colsize)**

**{**

**fstream newfile;**

**string record;**

**newfile.open("level1.txt", ios::in);**

**{**

**for (int row = 0; row < rowsize; row++)**

**{**

**getline(newfile, record);**

**for (int col = 0; col < colsize; col++)**

**{**

**maze[row][col] = record[col];**

**}**

**};**

**}**

**}**

**void load\_level\_2(char maze[][107])**

**{**

**fstream newfile;**

**string record;**

**newfile.open("level2.txt", ios::in);**

**{**

**for (int row = 0; row < 22; row++)**

**{**

**getline(newfile, record);**

**for (int col = 0; col < 107; col++)**

**{**

**maze[row][col] = record[col];**

**}**

**};**

**}**

**}**

**void print\_level2\_maze(char maze[][107])**

**{**

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

**SetConsoleTextAttribute(hConsole, 10);**

**for (int i = 0; i < 22; i++)**

**{**

**for (int j = 0; j < 107; j++)**

**{**

**cout << maze[i][j];**

**}**

**cout << endl;**

**}**

**}**

**// Function to print something on the specific loaction**

**void gotoxy(int x, int y)**

**{**

**COORD coordinates;**

**coordinates.X = x;**

**coordinates.Y = y;**

**SetConsoleCursorPosition(GetStdHandle(STD\_OUTPUT\_HANDLE), coordinates);**

**}**

**// Function to print on the maze**

**void print(char maze[][107])**

**{**

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

**SetConsoleTextAttribute(hConsole, 10);**

**for (int i = 0; i < 24; i++)**

**{**

**for (int j = 0; j < 107; j++)**

**{**

**cout << maze[i][j];**

**}**

**cout << endl;**

**}**

**}**

**// Function to print BALL B on the maze to the left side**

**void movefirst\_p\_Left(char maze[][107], int &B\_X, int &B\_Y, int &score, int &health)**

**{**

**if (maze[B\_X][B\_Y - 1] == ' ' || maze[B\_X][B\_Y - 1] == '\*')**

**{**

**// put the space on the location of B**

**maze[B\_X][B\_Y] = ' ';**

**gotoxy(B\_Y, B\_X);**

**cout << " ";**

**if (maze[B\_X][B\_Y - 1] == '\*')**

**{**

**calculate\_score(score);**

**}**

**B\_Y = B\_Y - 1;**

**if (maze[B\_X][B\_Y - 1] == 'o')**

**{**

**calculate\_health(health);**

**}**

**if (maze[B\_X][B\_Y - 1] == 'E')**

**{**

**score = score + 10;**

**}**

**// print the player B**

**gotoxy(B\_Y, B\_X);**

**cout << "B";**

**}**

**}**

**// Function to print BALL B on the maze to the right side**

**void movefirst\_p\_Right(char maze[][107], int &B\_X, int &B\_Y, int &score, int &health)**

**{**

**if (maze[B\_X][B\_Y + 1] == ' ' || maze[B\_X][B\_Y + 1] == '\*')**

**{**

**// put the space on the location of B**

**maze[B\_X][B\_Y] = ' ';**

**gotoxy(B\_Y, B\_X);**

**cout << " ";**

**if (maze[B\_X][B\_Y + 1] == '\*')**

**{**

**calculate\_score(score);**

**}**

**B\_Y = B\_Y + 1;**

**if (maze[B\_X][B\_Y + 1] == 'o')**

**{**

**calculate\_health(health);**

**}**

**if (maze[B\_X][B\_Y + 1] == 'E')**

**{**

**score = score + 10;**

**}**

**// if right of the ball is space then ball will fall down**

**if (maze[B\_X][B\_Y + 1] == ' ' && maze[B\_X - 1][B\_Y] == ' ')**

**{**

**movefirst\_p\_Down(maze, B\_X, B\_Y, score, health);**

**}**

**gotoxy(B\_Y, B\_X);**

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

**SetConsoleTextAttribute(hConsole, 7);**

**cout << "B";**

**}**

**} // Function to print BALL B on the maze to the up side**

**void movefirst\_p\_Up(char maze[][107], int &B\_X, int &B\_Y, int &score, int &health)**

**{**

**if (maze[B\_X - 1][B\_Y] == ' ' || maze[B\_X - 1][B\_Y] == '\*')**

**{**

**// put the space on the location of B**

**maze[B\_X][B\_Y] = ' ';**

**gotoxy(B\_Y, B\_X);**

**cout << " ";**

**if (maze[B\_X - 1][B\_Y] == '\*')**

**{**

**calculate\_score(score);**

**}**

**B\_X = B\_X - 1;**

**if (maze[B\_X - 1][B\_Y] == 'o')**

**{**

**calculate\_health(health);**

**}**

**if (maze[B\_X - 1][B\_Y] == 'E')**

**{**

**score = score + 10;**

**}**

**// if there is the dot on the loctaion of B the score increases by 1**

**gotoxy(B\_Y, B\_X);**

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

**SetConsoleTextAttribute(hConsole, 7);**

**cout << "B";**

**}**

**}**

**// Function to print BALL B on the maze to the down side**

**void movefirst\_p\_Down(char maze[][107], int &B\_X, int &B\_Y, int &score, int &health)**

**{**

**if (maze[B\_X + 1][B\_Y] == ' ' || maze[B\_X + 1][B\_Y] == '\*')**

**{**

**// put the space on the location of B**

**maze[B\_X][B\_Y] = ' ';**

**gotoxy(B\_Y, B\_X);**

**cout << " ";**

**if (maze[B\_X + 1][B\_Y] == '\*')**

**{**

**calculate\_score(score);**

**}**

**do**

**{**

**B\_X = B\_X + 1;**

**} while (maze[B\_X + 1][B\_Y] == ' ');**

**if (maze[B\_X + 1][B\_Y] == 'o')**

**{**

**calculate\_health(health);**

**}**

**if (maze[B\_X + 1][B\_Y] == 'E')**

**{**

**score = score + 10;**

**}**

**gotoxy(B\_Y, B\_X);**

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

**SetConsoleTextAttribute(hConsole, 7);**

**cout << "B";**

**}**

**}**

**void bounce\_ball\_right(char maze[][107], int &B\_X, int &B\_Y, int &score, int &health)**

**{**

**for (int x = 0; x < 5; x++)**

**{**

**Sleep(60);**

**movefirst\_p\_Up(maze, B\_X, B\_Y, score, health);**

**}**

**for (int x = 0; x < 2; x++)**

**{**

**Sleep(60);**

**movefirst\_p\_Right(maze, B\_X, B\_Y, score, health);**

**}**

**for (int x = 0; x < 6; x++)**

**{**

**Sleep(60);**

**movefirst\_p\_Down(maze, B\_X, B\_Y, score, health);**

**Sleep(100);**

**}**

**}**

**void bounce\_ball\_left(char maze[][107], int &B\_X, int &B\_Y, int &score, int &health)**

**{**

**for (int x = 0; x < 5; x++)**

**{**

**Sleep(20);**

**movefirst\_p\_Up(maze, B\_X, B\_Y, score, health);**

**}**

**for (int x = 0; x < 2; x++)**

**{**

**Sleep(20);**

**movefirst\_p\_Left(maze, B\_X, B\_Y, score, health);**

**}**

**for (int x = 0; x < 6; x++)**

**{**

**Sleep(20);**

**movefirst\_p\_Down(maze, B\_X, B\_Y, score, health);**

**Sleep(100);**

**}**

**}**

**// Fnuction to monve sppider**

**void movemet\_of\_spider1(char maze[][107], string &direction, int &rowCount, int &colIdx, int &health, int &B\_X, int &B\_Y, char previouitem1)**

**{ // Sleep(100);**

**if (direction == "down")**

**{**

**if (maze[rowCount + 1][colIdx] == '#' || maze[rowCount + 1][colIdx] == '%' || maze[rowCount + 1][colIdx] == '|' || maze[rowCount + 1][colIdx] == '\*' || maze[rowCount + 1][colIdx] == 'S')**

**{**

**direction = "up";**

**}**

**else**

**{**

**maze[rowCount][colIdx] = previouitem1;**

**gotoxy(colIdx, rowCount);**

**cout << previouitem1;**

**// increment in row to print on the +1 index**

**rowCount++;**

**previouitem1 = maze[rowCount][colIdx];**

**gotoxy(colIdx, rowCount);**

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

**SetConsoleTextAttribute(hConsole, 4);**

**cout << 'S';**

**// To check ball index and spider index**

**// if both are at same index then -health**

**if (rowCount + 1 == B\_X && colIdx == B\_Y)**

**{**

**calculate\_health(health);**

**}**

**}**

**}**

**else if (direction == "up")**

**{**

**maze[rowCount][colIdx] = ' ';**

**gotoxy(colIdx, rowCount);**

**cout << " ";**

**rowCount--;**

**gotoxy(colIdx, rowCount);**

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

**SetConsoleTextAttribute(hConsole, 4);**

**cout << 'S';**

**// To check ball index and spider index**

**// if both are at same index then -health**

**if (rowCount - 1 == B\_X && colIdx == B\_Y)**

**{**

**calculate\_health(health);**

**}**

**if (maze[rowCount - 1][colIdx] == '#' || maze[rowCount - 1][colIdx] == '\*' || maze[rowCount - 1][colIdx] == 'S')**

**{**

**direction = "down";**

**}**

**}**

**}**

**void movemet\_of\_spider2(char maze[][107], string &direction2, int &rowCount2, int &colIdx2, int &B\_X, int &B\_Y, int &health)**

**{**

**if (direction2 == "down")**

**{**

**if (maze[rowCount2 + 1][colIdx2] == '#' || maze[rowCount2 + 1][colIdx2] == '%' || maze[rowCount2 + 1][colIdx2] == '|')**

**{**

**direction2 = "up";**

**}**

**else**

**{**

**gotoxy(colIdx2, rowCount2);**

**cout << " ";**

**rowCount2++;**

**gotoxy(colIdx2, rowCount2);**

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

**SetConsoleTextAttribute(hConsole, 4);**

**cout << 'S';**

**// To check ball index and spider index**

**// if both are at same index then -health**

**if (rowCount2 + 1 == B\_X && colIdx2 == B\_Y)**

**{**

**calculate\_health(health);**

**}**

**}**

**}**

**else if (direction2 == "up")**

**{**

**gotoxy(colIdx2, rowCount2);**

**cout << " ";**

**rowCount2--;**

**gotoxy(colIdx2, rowCount2);**

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

**SetConsoleTextAttribute(hConsole, 4);**

**cout << 'S';**

**// To check ball index and spider index**

**// if both are at same index then -health**

**if (rowCount2 - 1 == B\_X && colIdx2 == B\_Y)**

**{**

**calculate\_health(health);**

**}**

**if (maze[rowCount2 - 1][colIdx2] == '#')**

**{**

**direction2 = "down";**

**}**

**}**

**}**

**void movemet\_of\_spider3(char maze[][107], string &direction3, int &colCount, int &rowIdx, int &B\_X, int &B\_Y, int &health)**

**{**

**if (direction3 == "left")**

**{**

**if (maze[rowIdx][colCount + 1] != ' ')**

**{**

**direction3 = "right";**

**}**

**else**

**{**

**gotoxy(colCount, rowIdx);**

**cout << " ";**

**colCount++;**

**gotoxy(colCount, rowIdx);**

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

**SetConsoleTextAttribute(hConsole, 4);**

**cout << 'S';**

**// To check ball index and spider index**

**// if both are at same index then -health**

**if (rowIdx == B\_X && colCount + 1 == B\_Y)**

**{**

**calculate\_health(health);**

**}**

**}**

**}**

**if (direction3 == "right")**

**{**

**gotoxy(colCount, rowIdx);**

**cout << " ";**

**colCount--;**

**gotoxy(colCount, rowIdx);**

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

**SetConsoleTextAttribute(hConsole, 4);**

**cout**

**<< 'S';**

**// To check ball index and spider index**

**// if both are at same index then -health**

**if (rowIdx == B\_X && colCount - 1 == B\_Y)**

**{**

**calculate\_health(health);**

**}**

**if (maze[rowIdx][colCount - 1] != ' ')**

**{**

**direction3 = "left";**

**}**

**}**

**}**

**void fire\_X(char maze[][107], int &enemyX, int &enemyY, int &fire\_of\_X\_X, int &fire\_of\_X\_Y, int &B\_X, int &B\_Y, int &health)**

**{**

**if (maze[fire\_of\_X\_X][fire\_of\_X\_Y - 1] == ' ')**

**{ // movmenet from left to right decreases of bomb of enemy X**

**fire\_of\_X\_Y--;**

**// locate enemy X bomb**

**gotoxy(fire\_of\_X\_Y, fire\_of\_X\_X);**

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

**SetConsoleTextAttribute(hConsole, 4);**

**cout << ".";**

**// reset the orevious postion into blank(empty space)**

**gotoxy(fire\_of\_X\_Y + 1, fire\_of\_X\_X);**

**cout << " ";**

**// to bring back fire if reached to |**

**if (maze[fire\_of\_X\_X][fire\_of\_X\_Y - 1] == '|' || maze[fire\_of\_X\_X][fire\_of\_X\_Y - 1] == 'B')**

**{**

**// if fire reached the obstacle then reset the fire**

**gotoxy(fire\_of\_X\_Y, fire\_of\_X\_X);**

**cout << " ";**

**fire\_of\_X\_X = enemyX;**

**fire\_of\_X\_Y = enemyY;**

**}**

**if (fire\_of\_X\_X == B\_X && fire\_of\_X\_Y == B\_Y)**

**{**

**calculate\_health(health);**

**}**

**}**

**}**

**int random()**

**{**

**srand(time(0));**

**int value = 1 + (rand() % 4);**

**return value;**

**}**

**void random\_movement(char maze[][107], int &rowidx4, int &colidx4, char &previousItem3, int &B\_X, int &B\_Y, int &health)**

**{**

**int value = random();**

**if (value == 1)**

**{ // move right**

**if (maze[rowidx4][colidx4 + 1] == ' ')**

**{**

**gotoxy(colidx4, rowidx4);**

**cout << " ";**

**colidx4++;**

**if (rowidx4 == B\_X && colidx4 + 1 == B\_Y)**

**{**

**calculate\_health(health);**

**}**

**maze[rowidx4][colidx4] = ' ';**

**gotoxy(colidx4, rowidx4);**

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

**SetConsoleTextAttribute(hConsole, 4);**

**cout << 'g';**

**}**

**}**

**if (value == 2)**

**{ // move left**

**if (maze[rowidx4][colidx4 - 1] == ' ')**

**{**

**gotoxy(colidx4, rowidx4);**

**cout << " ";**

**colidx4--;**

**if (rowidx4 == B\_X && colidx4 - 1 == B\_Y)**

**{**

**calculate\_health(health);**

**}**

**maze[rowidx4][colidx4] = ' ';**

**gotoxy(colidx4, rowidx4);**

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

**SetConsoleTextAttribute(hConsole, 4);**

**cout << 'g';**

**}**

**}**

**if (value == 3)**

**{ // move up**

**if (maze[rowidx4 - 1][colidx4] == ' ')**

**{**

**gotoxy(colidx4, rowidx4);**

**cout << " ";**

**rowidx4--;**

**if (rowidx4 - 1 == B\_X && colidx4 == B\_Y)**

**{**

**calculate\_health(health);**

**}**

**maze[rowidx4][colidx4] = ' ';**

**gotoxy(colidx4, rowidx4);**

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

**SetConsoleTextAttribute(hConsole, 4);**

**cout << 'g';**

**}**

**}**

**if (value == 4)**

**{ // move down**

**if (maze[rowidx4 + 1][colidx4] == ' ')**

**{**

**gotoxy(colidx4, rowidx4);**

**cout << " ";**

**rowidx4++;**

**if (rowidx4 + 1 == B\_X && colidx4 == B\_Y)**

**{**

**calculate\_health(health);**

**}**

**maze[rowidx4][colidx4] = ' ';**

**gotoxy(colidx4, rowidx4);**

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

**SetConsoleTextAttribute(hConsole, 4);**

**cout << 'g';**

**}**

**}**

**}**

**void level\_changer(char maze[][107], int &B\_X, int &B\_Y, int &score, int &health, int &level , int &rowCount , int &colIdx , int &rowCount2 , int &colIdx2 ,int &colCount ,int &rowIdx , int &rowidx4 , int &colidx4 )**

**{ // in case of find L which is level changer**

**if (maze[B\_X + 1][B\_Y] == 'L' || maze[B\_X][B\_Y + 1] == 'L')**

**{**

**if (level == 1)**

**{**

**level = 2;**

**B\_X = 21;**

**B\_Y = 3;**

**// index for the movement of spider 1**

**rowCount = 2;**

**colIdx = 5;**

**// index for the movement of spder 2**

**rowCount2 = 2;**

**colIdx2 = 7;**

**// indes for the movement of sprder 3**

**colCount = 32;**

**rowIdx = 8;**

**// index for the random movement of spider4**

**rowidx4 = 19;**

**colidx4 = 90;**

**system("cls");**

**load\_level\_2(maze);**

**print\_level2\_maze(maze);**

**}**

**else**

**{**

**}**

**}**

**}**

**Student Reg. No. :**   **Student Name.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **A-Extensive Evidence** | **B-Convincing Evidence** | **C-Limited Evidence** | **D-No Evidence** |
| Documentation Formatting **Grade:** | All the documentation meets all the criteria. | Documentation is well 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. In case of board game (Pong, Ludo and 2048 is acceptable) | 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. |
| **Checked by:** | Click or tap here to enter text. |