**MAZE**



Session: 2021 – 2025

**Submitted by:**

Abdul Mateen       2021-CS-190

**Supervised by:**

Dr. Maida Shahid

Department of Computer Science

**University of Engineering and Technology**

**Lahore Pakistan**

# CONTENTS

[Short Description of Game: 3](#_gjdgxs)

Game Characters Description: 3

[Rules & Interactions:](#_1fob9te) 3

Goal of the Game3

Screenshot of the Game: 4

Data Structures: 7

Functions Prototype: 7

Complete Code: 9

# SHORT DESCRIPTION OF GAME

This is basically a 2D game made in C++ using loops, functions, file handling and two dimensional arrays. In this game player has to move the character to the home while avoiding the obstacles. Obstacles are enemies and rockets. Player has to pass at least 3 check points to open the home.

**Key Features:**

* Graphic user interface
* Endless moves
* Easy to control
* Smooth Game
* Leader Board to keep track of scores

# GAME CHARACTERS DESCRIPTION

There are 3 characters in game;

1. Players
2. Enemies
3. Rocket launchers

# RULES & INTERACTIONS

In game you have to move the player to its home. You have to avoid the enemies because they follow you and you have also have to avoid the rocket launchers which are fired the borders. The home is locked to open it you have to get at least three checkpoints.

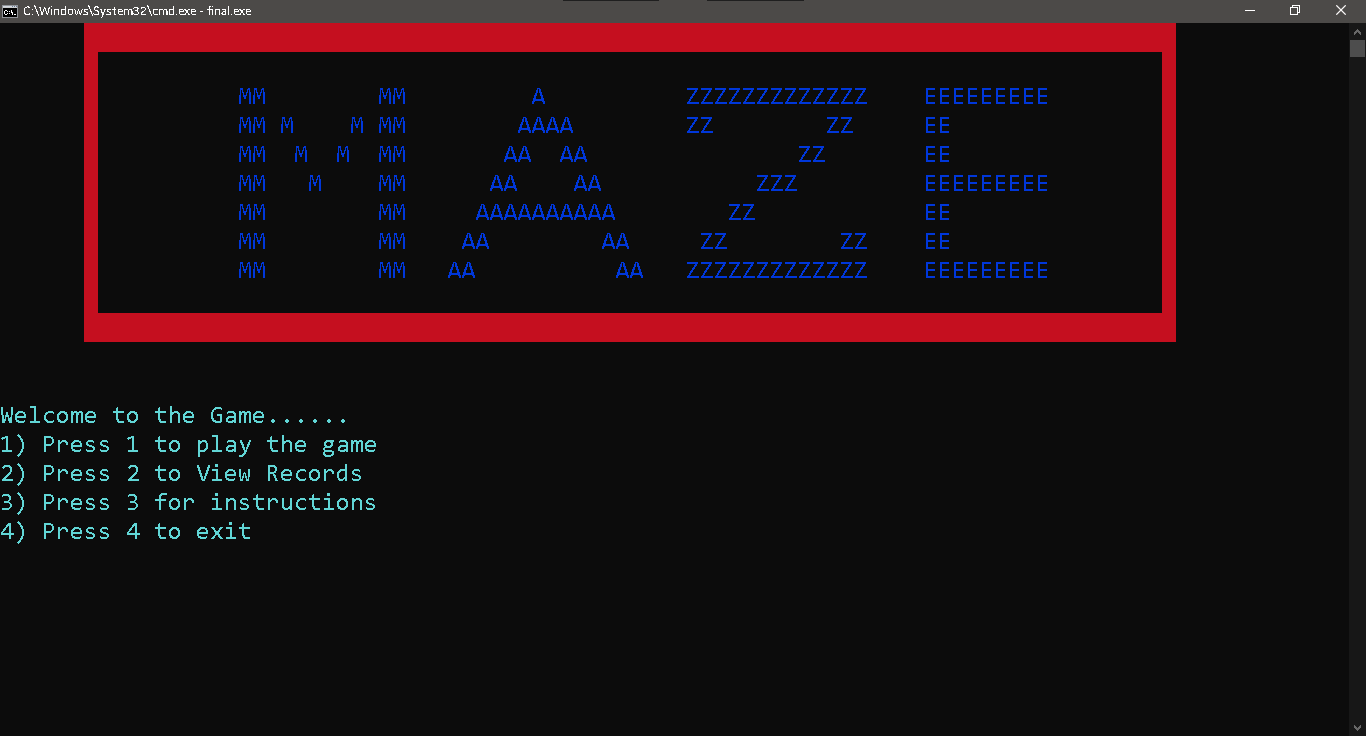
* Press Left arrow key for left movement
* Press Right arrow key for right movement
* Press Up arrow key for up movement
* Press Down arrow key for down movement

# GOAL

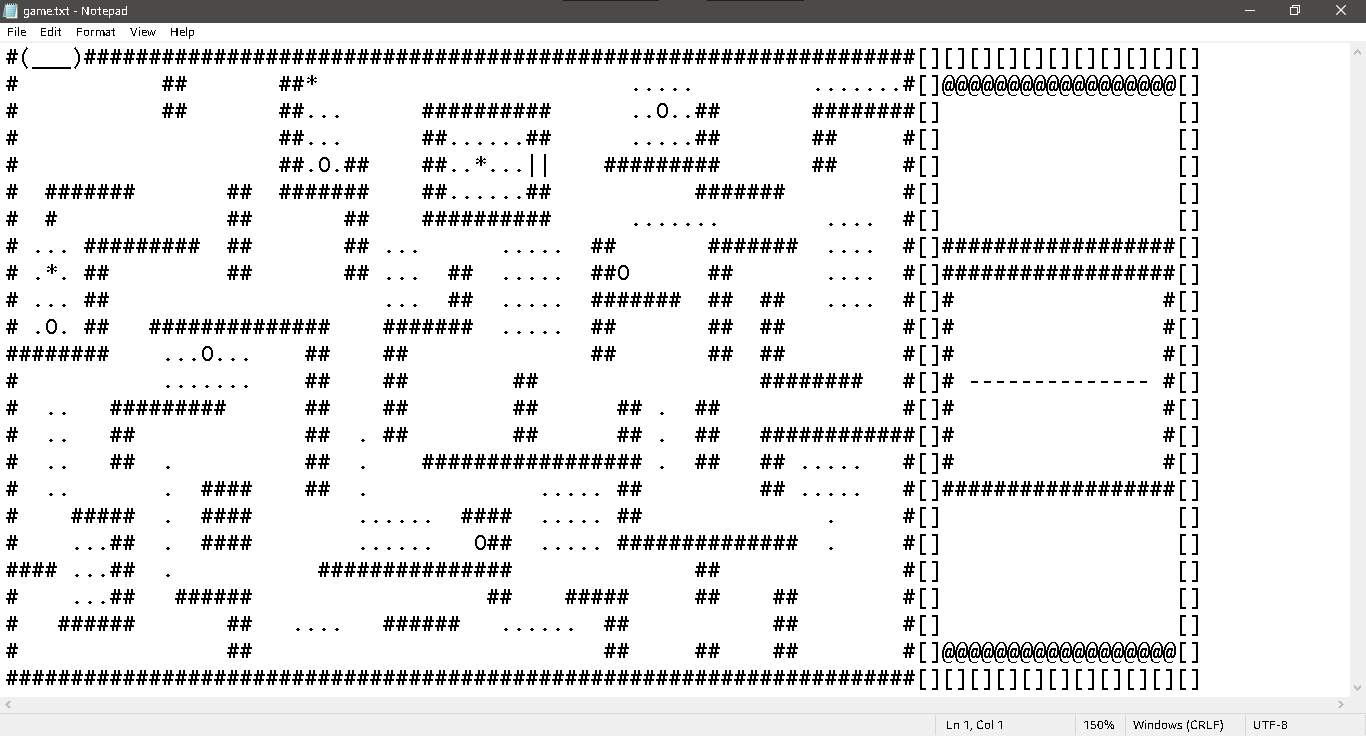
The goal of the game is to move the player to its home.

# SCREENSHOT OF GAME

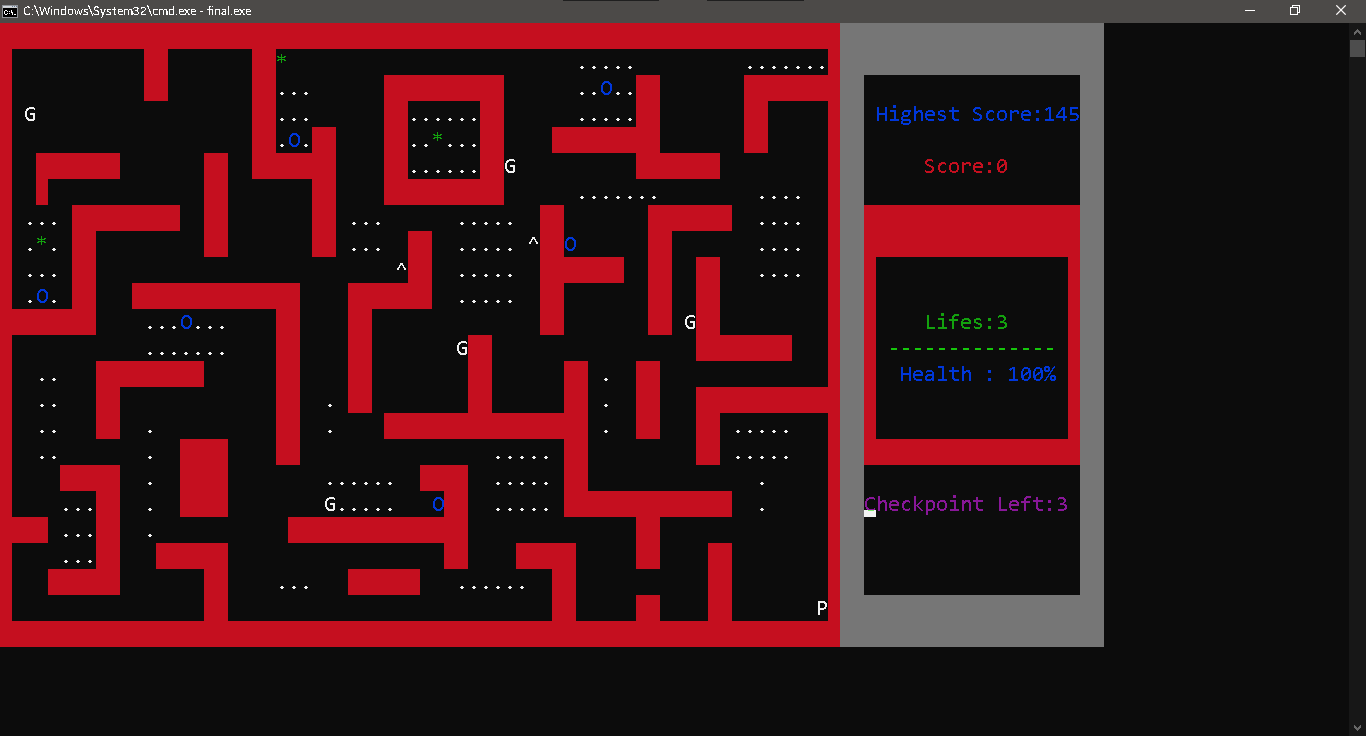
**Main Menu:**



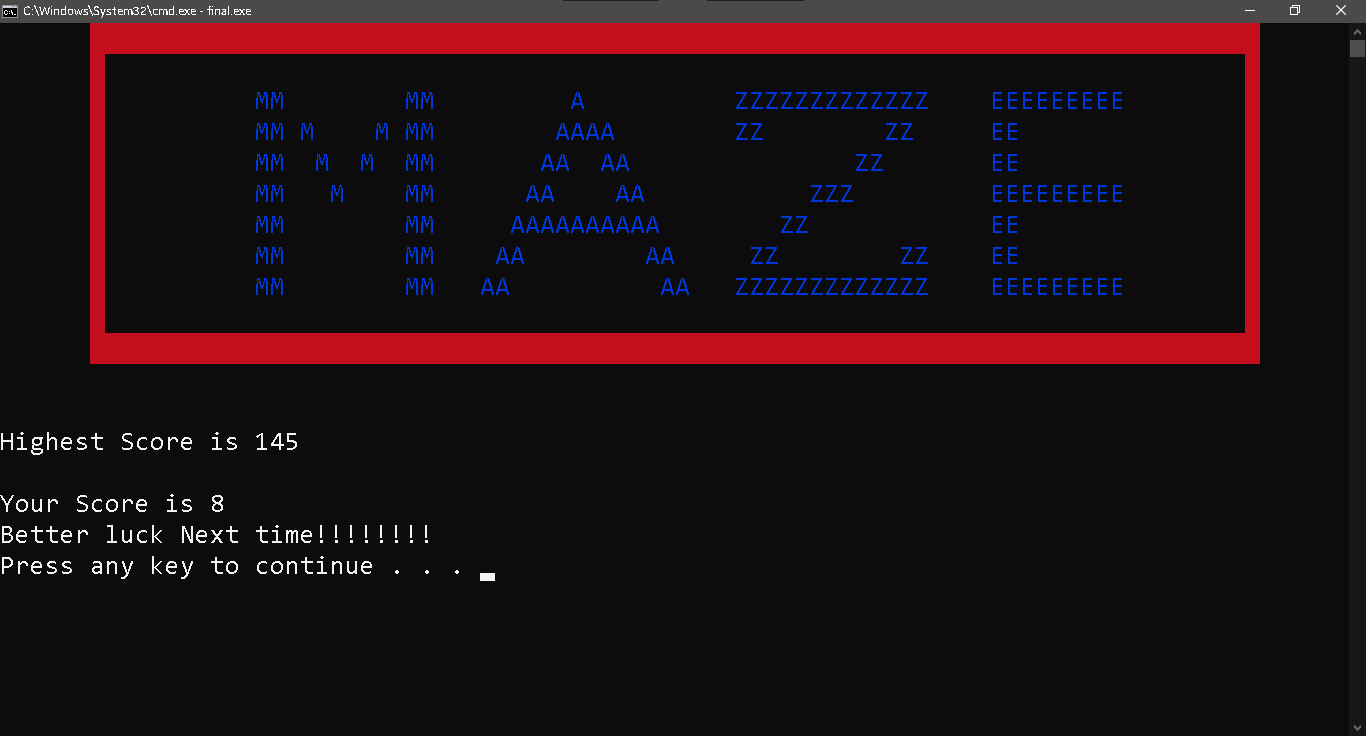
**Board of Game:**



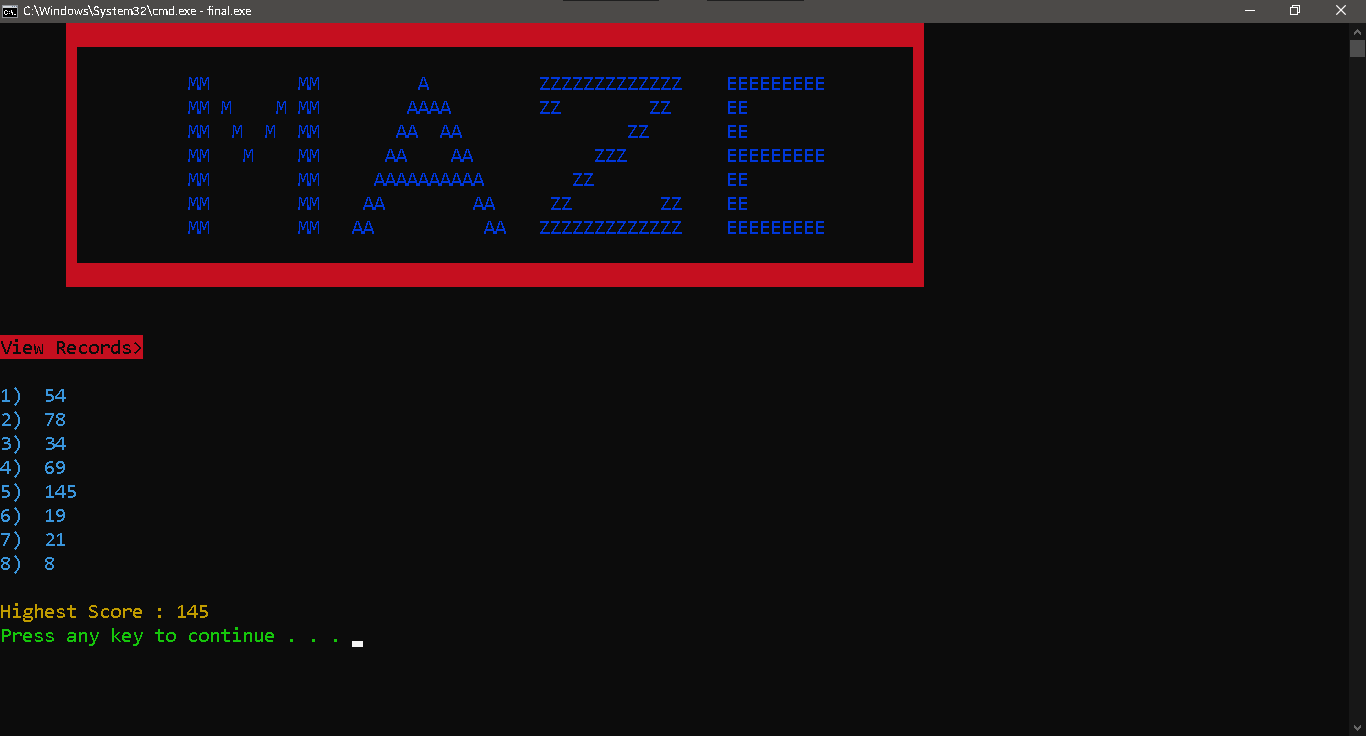
**Play Game:**



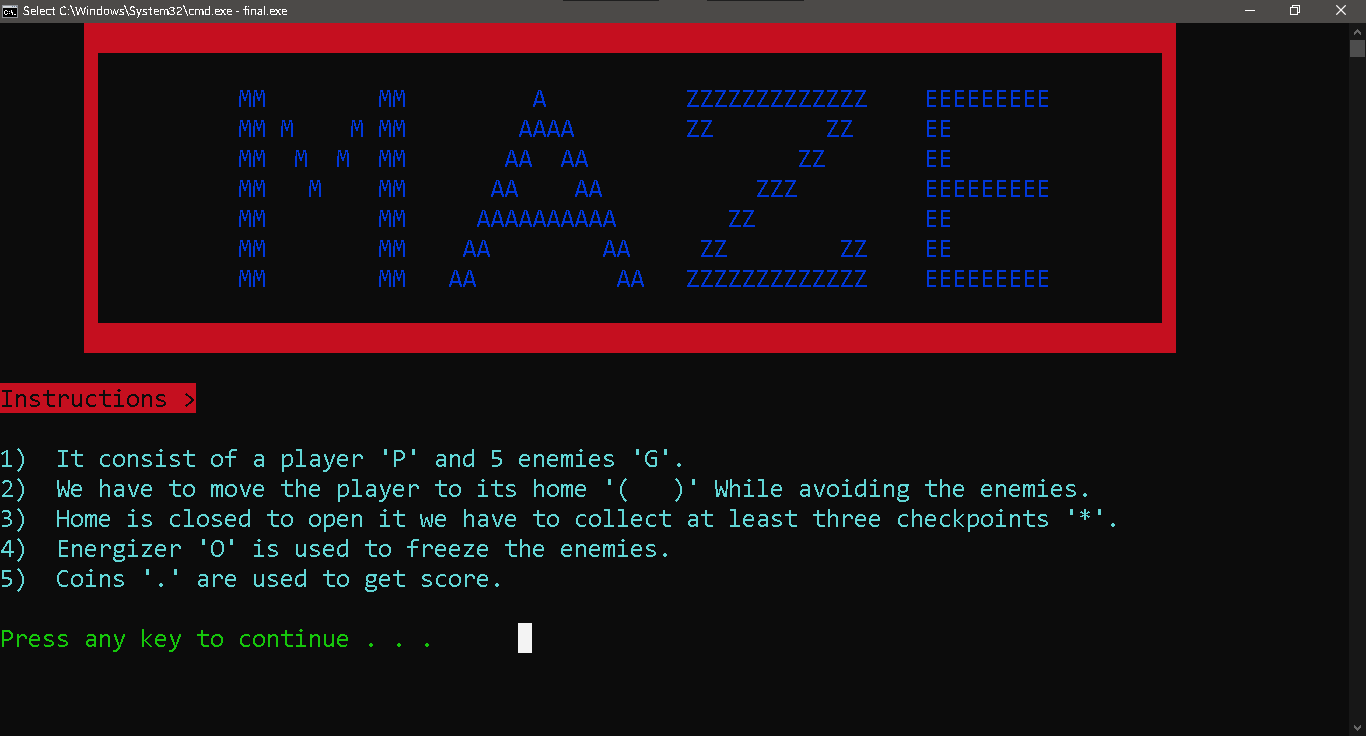
**Game Over:**

****

**Leader Board:**



**Instructions:**



**Quit Option:**



# DATA STRUCTURES

**Arrays:**

Int scores[20]={0};

Char maze[24][93];

Char header[11][93];

# FUNCTIONS PROTOTYPES

**//-----------for file handling----------------**

load\_maze();

load\_header();

storedata();

loadrecords();

viewrecords();

**//----------Display----------------------------**

header();

main\_menu();

display();

score1();

**//----------Rocket Launchers-------------**

fire1();

fire2();

fire3();

**//----------Enemies--------------------------**

ghostDirection1();

ghostDirection2();

ghostDirection3();

ghostDirection4();

ghostDirection5();

enemy1();

enemy2();

enemy3();

enemy4();

enemy5();

**//------------Player--------------------------**

moveup();

movedown();

moveright();

moveleft();

**//----------for concole setting-------------**

gotoxy(int x,int y);

void setcursor(bool visible,DWORD size);

**//----------instructions---------------------**

void instructions();

**//----------for score display --------------**

find\_highest();

store\_score\_array();

checkpoint();

**//---------Game End---------------**

game\_end\_score();

game\_esc();

gameend();

gameover();

# FULL CODE

#include<iostream>

#include<conio.h>

#include<windows.h>

#include<time.h>

#include<fstream>

#include<cmath>

#include<cstdlib>

#include<stdlib.h>

using namespace std;

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

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

COORD CursorPosition;

int X = 22, Y = 68;

int num\_record=0;

int score = 0, stars = 0;

bool running = true;

int highest\_score = 0;

char pre = ' ', pre2 = ' ', pre3 = ' ', pre4 = ' ', pre5 = ' ';

bool ft = 1, st = 0;

bool energizer = 0;

int gx1 = 9, gy1 = 43;

int gx2 = 3, gy2 = 3;

int gx3 = 16, gy3 = 27;

int gx4 = 6, gy4 = 55;

int gx5 = 9, gy5 = 9;

int movecount = 0, movecount2 = 0;

int energycount = 0;

int lifes = 3;

int health=100;

int fireX1 = 9, fireY1 = 10;

int fireX2 = 22, fireY2 = 22;

int fireX3 = 1, fireY3 = 44;

int firecount1 = 1, firecount2 = 0,firecount3=0;

bool restart1 = 0, restart2 = 0,restart3 = 0;

int scores[20] = {-1};

char maze[24][93];

char headerA[11][93];

void load\_maze()

{

fstream file;

string str;

int x=0;

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

for (int x = 0; x < 24; x++)

{

getline(file, str);

for (int m = 0; m < 93; m++)

{

maze[x][m] = str[m];

}

}

file.close();

}

void load\_header()

{

string str;

fstream file;

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

for(int m=0;m<12;m++)

{

getline(file,str);

for(int x=0;x<93;x++)

{

headerA[m][x]=str[x];

}

}

}

void header()

{

system("cls");

for(int x=0;x<11;x++)

{

for(int m=0;m<93;m++)

{

if(headerA[x][m] == '\_')

{

SetConsoleTextAttribute(color,68);

}

if(headerA[x][m] == '|')

{

SetConsoleTextAttribute(color,68);

}

if(headerA[x][m] == 'M' ||headerA[x][m] == 'A' || headerA[x][m] == 'Z' || headerA[x][m] == 'E' )

{

SetConsoleTextAttribute(color,1);

}

cout<<headerA[x][m];

SetConsoleTextAttribute(color,15);

}

cout<<endl;

}

}

void find\_highest()

{

for(int x=0;x<num\_record;x++)

{

if(scores[x] > highest\_score)

{

highest\_score = scores[x];

}

}

}

void main\_menu()

{

header();

SetConsoleTextAttribute(color,11);

cout<<endl<<endl<<"Welcome to the Game......"<<endl;

cout<<"1) Press 1 to play the game"<<endl;

cout<<"2) Press 2 to View Records "<<endl;

cout<<"3) Press 3 for instructions"<<endl;

cout<<"4) Press 4 to exit"<<endl;

}

void instructions()

{

header();

SetConsoleTextAttribute(color,64);

cout<<endl<<"Instructions >"<<endl<<endl;

fstream file;

string str;

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

while(!file.eof())

{

SetConsoleTextAttribute(color,11);

getline(file,str);

cout<<str<<endl;

}

file.close();

cout<<endl;

SetConsoleTextAttribute(color,10);

system("pause");

SetConsoleTextAttribute(color,15);

}

void gotoxy(int x,int y)

{

COORD c;

c.X = x;

c.Y = y;

SetConsoleCursorPosition(

GetStdHandle(STD\_OUTPUT\_HANDLE), c);

}

void storedata()

{

fstream file;

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

for(int m=0;m<24;m++)

{

for(int x=0 ;x<93;x++)

{

file<<maze[m][x];

}

file<<endl;

}

file.close();

}

void store\_score\_array()

{

scores[num\_record]=score;

num\_record++;

}

void display()

{

for (int x=0;x< 24;x++)

{

for(int m=0;m<93;m++)

{

if(maze[x][m] == '#')

{

SetConsoleTextAttribute(color,68);

}

if(maze[x][m] == '|')

{

SetConsoleTextAttribute(color,68);

}

if(maze[x][m] == 'O')

{

SetConsoleTextAttribute(color,1);

}

if(maze[x][m] == '\*')

{

SetConsoleTextAttribute(color,2);

}

if(maze[x][m] == '.')

{

SetConsoleTextAttribute(color,15);

}

if(maze[x][m] == '\_')

{

SetConsoleTextAttribute(color,68);

}

if(maze[x][m] == '-')

{

SetConsoleTextAttribute(color,10);

}

if(maze[x][m] == ']' || maze[x][m] == '[')

{

SetConsoleTextAttribute(color,136);

}

if(maze[x][m] == '@')

{

SetConsoleTextAttribute(color,136);

}

if(maze[x][m] == '(' || maze[x][m] == ')')

{

SetConsoleTextAttribute(color,68);

}

cout<<maze[x][m];

SetConsoleTextAttribute(color,15);

}

cout<<endl;

}

SetConsoleTextAttribute(color,15);

}

void score1()

{

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

}

void checkpoint()

{

stars++;

if(stars == 2)

{

maze[4][41]=' ';

gotoxy(41,4);

cout<<' ';

maze[4][40]=' ';

gotoxy(40,4);

cout<<' ';

}

if(stars>=3)

{

maze[0][1]='(';

maze[0][2]=' ';

maze[0][3]=' ';

maze[0][4]=' ';

maze[0][5]=')';

gotoxy(1,0);

cout<<'(';

gotoxy(2,0);

cout<<' ';

gotoxy(3,0);

cout<<' ';

gotoxy(4,0);

cout<<' ';

gotoxy(5,0);

cout<<')';

}

}

void game\_end\_score()

{

if(score<highest\_score)

{

cout<<"Highest Score is "<<highest\_score<<endl<<endl;

cout<<"Your Score is "<<score<<endl;

cout<<"Better luck Next time!!!!!!!!"<<endl;

}

else

{

cout<<"You have created new High Score!!!!!!!"<<endl<<endl;

cout<<"New High Score is "<<score<<endl;

highest\_score=score;

}

system("pause");

}

void game\_esc()

{

running=false;

system("cls");

}

void gameend()

{

maze[X][Y]=' ';

gotoxy(Y,X);

cout<<' ';

maze[gx1][gy1]=' ';

gotoxy(gy1,gx1);

cout<<' ';

maze[gx2][gy2]=' ';

gotoxy(gy2,gx2);

cout<<' ';

maze[gx3][gy3]=' ';

gotoxy(gy3,gx3);

cout<<' ';

maze[gx4][gy4]=' ';

gotoxy(gy4,gx4);

cout<<' ';

running=false;

system("cls");

header();

cout<<endl<<endl;

}

int ghostDirection1()

{

srand(time(0));

int result = 1 + (rand() % 4);

return result;

}

int ghostDirection2()

{

srand(time(0));

int result = 1 + ((rand()) % 2);

return result;

}

int ghostDirection3()

{

srand(time(0));

int result = 1 + ((rand()) % 2);

return result;

}

int ghostDirection4()

{

if(movecount == 4)

{

srand(time(0)) ;

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

movecount=0;

return value;

}

movecount++;

int xaxis=X-gx4;

int yaxis=Y-gy4;

int absx=abs(xaxis);

int absy=abs(yaxis);

if(absx > absy)

{

if(xaxis < 0)

return 1;

else

return 2;

}

else

{

if(yaxis < 0)

return 4;

else

return 3;

}

}

int ghostDirection5()

{

if(movecount2 == 3)

{

srand(time(0)) ;

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

movecount2=0;

return value;

}

movecount2++;

int xaxis=X-gx5;

int yaxis=Y-gy5;

int absx=abs(xaxis);

int absy=abs(yaxis);

if(absx > absy)

{

if(xaxis < 0)

return 1;

else

return 2;

}

else

{

if(yaxis < 0)

return 4;

else

return 3;

}

}

void gameover()

{

if (lifes == 1)

{

gameend();

game\_end\_score();

store\_score\_array();

}

else

{

health=100;

maze[X][Y]=' ';

gotoxy(Y,X);

cout<<' ';

lifes--;

X=22;

Y=68;

maze[X][Y]='P';

gotoxy(Y,X);

cout<<'P';

}

}

void healthover()

{

if (health == 25)

{

gameover();

}

else

{

health=health-25;

}

}

void moveup()

{

if (maze[X-1][Y] == '.')

{

score++;

}

if (maze[X-1][Y] == '\*')

{

score=score+10;

checkpoint();

}

if(energizer &&( maze[X - 1][Y] == 'G'|| maze[X - 1][Y] == '^'))

{

gotoxy(Y, X - 1);

cout << 'P';

maze[X - 1][Y] = 'P';

gotoxy(Y, X);

cout << ' ';

maze[X][Y] = ' ';

}

if(!energizer &&maze[X - 1][Y] == 'G')

{gameover();}

if(!energizer &&maze[X - 1][Y] == '^')

{healthover();}

if (maze[X - 1][Y] == ' ' || maze[X - 1][Y] == '.' || maze[X - 1][Y] == 'O' || maze[X - 1][Y] == 'P' || maze[X - 1][Y] == '\*')

{

if (maze[X - 1][Y] == 'O')

{

energizer = 1;

energycount=0;

}

gotoxy(Y, X - 1);

cout << 'P';

maze[X - 1][Y] = 'P';

gotoxy(Y, X);

cout << ' ';

maze[X][Y] = ' ';

X--;

}

}

void movedown()

{

if (maze[X+1][Y] == '.')

{

score++;

}

if (maze[X+1][Y] == '\*')

{

score=score+10;

checkpoint();

}

if(energizer &&( maze[X+1][Y] == 'G' || maze[X+1][Y] == '^'))

{

gotoxy(Y,X+1);

cout<<'P';

maze[X+1][Y]='P';

gotoxy(Y,X);

cout<<' ';

maze[X][Y]=' ';

}

if(!energizer && maze[X+1][Y] == 'G' )

{

gameover();

}

if(!energizer && maze[X+1][Y] == '^')

{

healthover();

}

if(maze[X+1][Y] == ' ' || maze[X+1][Y] == '.' || maze[X+1][Y] == 'O' || maze[X+1][Y] == 'P' || maze[X+1][Y] == '\*')

{

if(maze[X+1][Y] == 'O')

{energizer=1;

energycount=0;}

gotoxy(Y,X+1);

cout<<'P';

maze[X+1][Y]='P';

gotoxy(Y,X);

cout<<' ';

maze[X][Y]=' ';

X++;

}

}

void moveright()

{

if (maze[X][Y+1] == '.')

{

score++;

}

if (maze[X][Y+1] == '\*')

{

score=score+10;

checkpoint();

}

if(energizer && (maze[X][Y+1] == 'G' || maze[X][Y+1] == '^'))

{

gotoxy(Y+1,X);

cout<<'P';

maze[X][Y+1]='P';

gotoxy(Y,X);

cout<<' ';

maze[X][Y]=' ';

}

if(!energizer &&maze[X][Y+1] == 'G')

{

gameover();

}

if(!energizer && maze[X][Y+1] == '^')

{

healthover();

}

if(maze[X][Y+1] == ' ' || maze[X][Y+1] == '.' || maze[X][Y+1] == 'O' || maze[X][Y+1] == 'P' || maze[X][Y+1] == '\*')

{

if(maze[X][Y+1] == 'O')

{energizer=1;

energycount=0;}

gotoxy(Y+1,X);

cout<<'P';

maze[X][Y+1]='P';

gotoxy(Y,X);

cout<<' ';

maze[X][Y]=' ';

Y++;

}

}

void moveleft()

{

if (maze[X][Y - 1] == '.')

{

score++;

}

if (maze[X][Y - 1] == '\*')

{

score=score+10;

checkpoint();

}

if(energizer && (maze[X][Y - 1] == 'G' || maze[X][Y - 1] == '^'))

{

gotoxy(Y-1,X);

cout<<'P';

maze[X][Y-1]='P';

gotoxy(Y,X);

cout<<' ';

maze[X][Y]=' ';

}

if(!energizer &&maze[X][Y - 1] == 'G')

{

gameover();

}

if(!energizer && maze[X][Y - 1] == '^')

{

healthover();

}

if (maze[X][Y - 1] == ' ' || maze[X][Y - 1] == '.' || maze[X][Y - 1] == 'O' || maze[X][Y - 1] == 'P' || maze[X][Y - 1] == '\*')

{

if(maze[X][Y - 1] == 'O')

{energizer=1;

energycount=0;}

gotoxy(Y-1,X);

cout<<'P';

maze[X][Y-1]='P';

gotoxy(Y,X);

cout<<' ';

maze[X][Y]=' ';

Y--;

}

}

void enemy1()

{

int value = ghostDirection1();

if (value == 1)

{

if (maze[gx1 - 1][gy1] == ' ' || maze[gx1 - 1][gy1] == '.' || maze[gx1 - 1][gy1] == 'P'|| maze[gx1 - 1][gy1] == '\*')

{

maze[gx1][gy1] = pre;

gotoxy(gy1, gx1);

if (maze[gx1][gy1] == '.')

{

cout << '.';

}

else

{

cout << ' ';

}

gx1--;

if (maze[gx1][gy1] != 'P')

{

pre = maze[gx1][gy1];

}

if (maze[gx1][gy1] == 'P')

{

pre = ' ';

gameover();

}

maze[gx1][gy1] = 'G';

gotoxy(gy1, gx1);

cout << 'G';

}

}

if (value == 2)

{

if (maze[gx1 + 1][gy1] == ' ' || maze[gx1 + 1][gy1] == '.' || maze[gx1 + 1][gy1] == 'P' || maze[gx1 + 1][gy1] == '\*')

{

maze[gx1][gy1] = pre;

gotoxy(gy1, gx1);

if (maze[gx1][gy1] == '.')

{

cout << '.';

}

else

{

cout << ' ';

}

gx1++;

if (maze[gx1][gy1] != 'P') pre = maze[gx1][gy1];

if (maze[gx1][gy1] == 'P')

{

pre=' ';

gameover();

}

maze[gx1][gy1] = 'G';

gotoxy(gy1, gx1);

cout << 'G';

}

}

if (value == 3)

{

if (maze[gx1][gy1 + 1] == ' ' || maze[gx1][gy1 + 1] == '.' || maze[gx1][gy1 + 1] == 'P' || maze[gx1][gy1 + 1] == '\*')

{

maze[gx1][gy1] = pre;

gotoxy(gy1, gx1);

if (maze[gx1][gy1] == '.')

{

cout << '.';

}

else

{

cout << ' ';

}

gy1++;

if (maze[gx1][gy1] != 'P')

{

pre = maze[gx1][gy1];

}

if (maze[gx1][gy1] == 'P')

{

pre=' ';

gameover();

}

maze[gx1][gy1] = 'G';

gotoxy(gy1, gx1);

cout << 'G';

}

}

if (value == 4)

{

if (maze[gx1][gy1 - 1] == ' ' || maze[gx1][gy1 - 1] == '.' || maze[gx1][gy1 - 1] == 'P'|| maze[gx1][gy1 - 1] == '\*')

{

maze[gx1][gy1] = pre;

gotoxy(gy1, gx1);

if (maze[gx1][gy1] == '.')

{

cout << '.';

}

else

{

cout << ' ';

}

gy1--;

if (maze[gx1][gy1] != 'P') pre = maze[gx1][gy1];

if (maze[gx1][gy1] == 'P')

{

pre=' ';

gameover();

}

maze[gx1][gy1] = 'G';

gotoxy(gy1, gx1);

cout << 'G';

}

}

}

void enemy2()

{

int value = ghostDirection2();

if (value == 1)

{

if (maze[gx2][gy2 + 1] == ' ' || maze[gx2][gy2 + 1] == '.' || maze[gx2][gy2 + 1] == 'P')

{

maze[gx2][gy2] = pre2;

gotoxy(gy2, gx2);

if (maze[gx2][gy2] == '.')

{

cout << '.';

}

else

{

cout << ' ';

}

gy2++;

if (maze[gx2][gy2] != 'P')

{pre2 = maze[gx2][gy2];}

if (maze[gx2][gy2] == 'P')

{

pre2 =' ';

gameover();

}

maze[gx2][gy2] = 'G';

gotoxy(gy2, gx2);

cout << 'G';

}

}

if (value == 2)

{

if (maze[gx2][gy2 - 1] == ' ' || maze[gx2][gy2 - 1] == '.' || maze[gx2][gy2 - 1] == 'P')

{

maze[gx2][gy2] = pre2;

gotoxy(gy2, gx2);

if (maze[gx2][gy2] == '.')

{

cout << '.';

}

else

{

cout << ' ';

}

gy2--;

if (maze[gx2][gy2] != 'P')

{pre2 = maze[gx2][gy2];}

if (maze[gx2][gy2] == 'P')

{

pre2 =' ';

gameover();

}

maze[gx2][gy2] = 'G';

gotoxy(gy2, gx2);

cout << 'G';

}

}

}

void enemy3()

{

int value = ghostDirection3();

if (value == 1)

{

if (maze[gx3 - 1][gy3] == ' ' || maze[gx3 - 1][gy3] == '.' || maze[gx3 - 1][gy3] == 'P')

{

maze[gx3][gy3] = pre3;

gotoxy(gy3, gx3);

if (maze[gx3][gy3] == '.')

{

cout << '.';

}

else

{

cout << ' ';

}

gx3--;

if (maze[gx3][gy3] != 'P') pre3 = maze[gx3][gy3];

if (maze[gx3][gy3] == 'P')

{

pre3 =' ';

gameover();

}

maze[gx3][gy3] = 'G';

gotoxy(gy3, gx3);

cout << 'G';

}

}

if (value == 2)

{

if (maze[gx3 + 1][gy3] == ' ' || maze[gx3 + 1][gy3] == '.' || maze[gx3 + 1][gy3] == 'P')

{

maze[gx3][gy3] = pre3;

gotoxy(gy3, gx3);

if (maze[gx3][gy3] == '.')

{

cout << '.';

}

else

{

cout << ' ';

}

gx3++;

if(maze[gx3][gy3] != 'P') {pre3 = maze[gx3][gy3];}

if (maze[gx3][gy3] == 'P')

{

pre3 =' ';

gameover();

}

maze[gx3][gy3] = 'G';

gotoxy(gy3, gx3);

cout << 'G';

}

}

}

void enemy4()

{

int value = ghostDirection4();

if (value == 1)

{

if (maze[gx4 - 1][gy4] == ' ' || maze[gx4 - 1][gy4] == '.' || maze[gx4 - 1][gy4] == 'P')

{

maze[gx4][gy4] = pre4;

gotoxy(gy4, gx4);

if (maze[gx4][gy4] == '.')

{

cout << '.';

}

else

{

cout << ' ';

}

gx4--;

if (maze[gx4][gy4] != 'P')

{pre4 = maze[gx4][gy4];}

if (maze[gx4][gy4] == 'P')

{

pre4=' ';

gameover();

}

maze[gx4][gy4] = 'G';

gotoxy(gy4, gx4);

cout << 'G';

}

}

if (value == 2)

{

if (maze[gx4 + 1][gy4] == ' ' || maze[gx4 + 1][gy4] == '.' || maze[gx4 + 1][gy4] == 'P')

{

maze[gx4][gy4] = pre4;

gotoxy(gy4, gx4);

if (maze[gx4][gy4] == '.')

{

cout << '.';

}

else

{

cout << ' ';

}

gx4++;

if (maze[gx4][gy4] != 'P')

{pre4 = maze[gx4][gy4];}

if (maze[gx4][gy4] == 'P')

{

pre4=' ';

gameover();

}

maze[gx4][gy4] = 'G';

gotoxy(gy4, gx4);

cout << 'G';

}

}

if (value == 3)

{

if (maze[gx4][gy4 + 1] == ' ' || maze[gx4][gy4 + 1] == '.' || maze[gx4][gy4 + 1] == 'P')

{

maze[gx4][gy4] = pre4;

gotoxy(gy4, gx4);

if (maze[gx4][gy4] == '.')

{

cout << '.';

}

else

{

cout << ' ';

}

gy4++;

if (maze[gx4][gy4] != 'P')

{pre4 = maze[gx4][gy4];}

if (maze[gx4][gy4] == 'P')

{

pre4=' ';

gameover();

}

maze[gx4][gy4] = 'G';

gotoxy(gy4, gx4);

cout << 'G';

}

}

if (value == 4)

{

if (maze[gx4][gy4 - 1] == ' ' || maze[gx4][gy4 - 1] == '.' || maze[gx4][gy4 - 1] == 'P')

{

maze[gx4][gy4] = pre4;

gotoxy(gy4, gx4);

if (maze[gx4][gy4] == '.')

{

cout << '.';

}

else

{

cout << ' ';

}

gy4--;

if (maze[gx4][gy4] != 'P')

{pre4 = maze[gx4][gy4];}

if (maze[gx4][gy4] == 'P')

{

pre4=' ';

gameover();

}

maze[gx4][gy4] = 'G';

gotoxy(gy4, gx4);

cout << 'G';

}

}

}

void enemy5()

{

int value = ghostDirection5();

if (value == 1)

{

if (maze[gx5 - 1][gy5] == ' ' || maze[gx5 - 1][gy5] == '.' || maze[gx5 - 1][gy5] == 'P'|| maze[gx5 - 1][gy5] == 'O')

{

maze[gx5][gy5] = pre5;

gotoxy(gy5, gx5);

if (maze[gx5][gy5] == '.')

{

cout << '.';

}

else

{

cout << ' ';

}

gx5--;

if (maze[gx5][gy5] != 'P')

{pre5 = maze[gx5][gy5];}

if (maze[gx5][gy5] == 'P')

{

pre5=' ';

gameover();

}

maze[gx5][gy5] = 'G';

gotoxy(gy5, gx5);

cout << 'G';

}

}

if (value == 2)

{

if (maze[gx5 + 1][gy5] == ' ' || maze[gx5 + 1][gy5] == '.' || maze[gx5 + 1][gy5] == 'P' || maze[gx5 + 1][gy5] == 'O')

{

maze[gx5][gy5] = pre5;

gotoxy(gy5, gx5);

if (maze[gx5][gy5] == '.')

{

cout << '.';

}

else

{

cout << ' ';

}

gx5++;

if (maze[gx5][gy5] != 'P')

{pre5 = maze[gx5][gy5];}

if (maze[gx5][gy5] == 'P')

{

pre5=' ';

gameover();

}

maze[gx5][gy5] = 'G';

gotoxy(gy5, gx5);

cout << 'G';

}

}

if (value == 3)

{

if (maze[gx5][gy5 + 1] == ' ' || maze[gx5][gy5 + 1] == '.' || maze[gx5][gy5 + 1] == 'P'|| maze[gx5][gy5 + 1] == 'O')

{

maze[gx5][gy5] = pre5;

gotoxy(gy5, gx5);

if (maze[gx5][gy5] == '.')

{

cout << '.';

}

else

{

cout << ' ';

}

gy5++;

if (maze[gx5][gy5] != 'P')

{pre5 = maze[gx5][gy5];}

if (maze[gx5][gy5] == 'P')

{

pre5=' ';

gameover();

}

maze[gx5][gy5] = 'G';

gotoxy(gy5, gx5);

cout << 'G';

}

}

if (value == 4)

{

if (maze[gx5][gy5 - 1] == ' ' || maze[gx5][gy5 - 1] == '.' || maze[gx5][gy5 - 1] == 'P'|| maze[gx5][gy5 - 1] == 'O')

{

maze[gx5][gy5] = pre5;

gotoxy(gy5, gx5);

if (maze[gx5][gy5] == '.')

{

cout << '.';

}

else

{

cout << ' ';

}

gy5--;

if (maze[gx5][gy5] != 'P')

{pre5 = maze[gx5][gy5];}

if (maze[gx5][gy5] == 'P')

{

pre5=' ';

gameover();

}

maze[gx5][gy5] = 'G';

gotoxy(gy5, gx5);

cout << 'G';

}

}

}

void loadrecords()

{

num\_record=0;

string str;

fstream file;

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

for(int x=0;!file.eof();x++)

{

file>>scores[x];

num\_record++;

}

file.close();

find\_highest();

}

void viewrecords()

{

for(int x=0;x<num\_record;x++)

{

SetConsoleTextAttribute(color,3);

cout<<x+1<<") "<<scores[x]<<endl;

}

SetConsoleTextAttribute(color,6);

cout<<endl<<"Highest Score : "<<highest\_score<<endl;

SetConsoleTextAttribute(color,10);

system("pause");

SetConsoleTextAttribute(color,15);

}

void exitgame()

{

header();

cout<<endl<<endl;

SetConsoleTextAttribute(color,11);

cout<<"........................................................................................."<<endl;

cout<<" Thanks For Playing the Game"<<endl;

cout<<"........................................................................................."<<endl<<endl;

SetConsoleTextAttribute(color,10);

system("pause");

SetConsoleTextAttribute(color,15);

}

void store\_score\_file()

{

fstream file;

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

for(int x=0;x<num\_record;x++)

{

file<<endl<<scores[x];

}

file.close();

}

void fire1()

{

firecount1++;

if(firecount1==3)

{

firecount1=0;

if(maze[fireX1][fireY1+1] == '#' || maze[fireX1][fireY1+1] == 'G' )

{

maze[fireX1][fireY1]=' ';

gotoxy(fireY1,fireX1);

cout<<' ';

restart1=1;

}

else

{

if(maze[fireX1][fireY1+1] == 'P')

{

healthover();

}

maze[fireX1][fireY1]=' ';

gotoxy(fireY1,fireX1);

cout<<' ';

maze[fireX1][fireY1+1]='^';

gotoxy(fireY1+1,fireX1);

cout<<'^';

}

if (restart1)

{

fireX1 = 9, fireY1 = 10;

restart1=0;

}

fireY1++;

}

}

void fire2()

{

firecount2++;

if(firecount2==3)

{

firecount2=0;

if(maze[fireX2-1][fireY2] == '#' || maze[fireX2-1][fireY2] == 'G' )

{

maze[fireX2][fireY2]=' ';

gotoxy(fireY2,fireX2);

cout<<' ';

restart2=1;

}

else

{

if(maze[fireX2-1][fireY2] == 'P')

{

healthover();

}

maze[fireX2][fireY2]=' ';

gotoxy(fireY2,fireX2);

cout<<' ';

maze[fireX2-1][fireY2]='^';

gotoxy(fireY2,fireX2-1);

cout<<'^';

}

if (restart2)

{

fireX2 = 23, fireY2 = 22;

restart2=0;

}

fireX2--;

}

}

void fire3()

{

firecount3++;

if(firecount3==3)

{

firecount3=0;

if(maze[fireX3+1][fireY3] == '#' || maze[fireX3+1][fireY3] == 'G' )

{

maze[fireX3][fireY3]=' ';

gotoxy(fireY3,fireX3);

cout<<' ';

restart3=1;

}

else

{

if(maze[fireX3+1][fireY3] == 'P')

{

healthover();

}

maze[fireX3][fireY3]=' ';

gotoxy(fireY3,fireX3);

cout<<' ';

maze[fireX3+1][fireY3]='^';

gotoxy(fireY3,fireX3+1);

cout<<'^';

}

if (restart3)

{

fireX3 =1, fireY3 = 44 ;

restart3=0;

}

fireX3++;

}

}

void display\_high\_score()

{

gotoxy(73, 3);

SetConsoleTextAttribute(color, 1);

cout << "Highest Score:" << highest\_score;

SetConsoleTextAttribute(color, 15);

}

void setcursor(bool visible, DWORD size)

{

if (size == 0)

size = 20;

CONSOLE\_CURSOR\_INFO lpCursor;

lpCursor.bVisible = visible;

lpCursor.dwSize = size;

SetConsoleCursorInfo(console, &lpCursor);

}

/////////////////////////////////////////////// MAIN /////////////////////////////////////////////////

main()

{

setcursor( 0, 0);

loadrecords();

load\_header();

char option=' ';

while(option!='4')

{

system("cls");

main\_menu();

cin>>option;

if(option=='1')

{

load\_maze();

gx1 = 9, gy1 = 43;

gx2 = 3, gy2 = 3;

gx3 = 16, gy3 = 27;

gx4 = 6, gy4 = 55;

gx5 = 9, gy5 = 9;

movecount = 0, movecount2 = 0;

energycount = 0;

lifes = 3;

X = 22, Y = 68;

score = 0, stars = 0;

running = true;

pre = ' ', pre2 = ' ', pre3 = ' ', pre4 = ' ', pre5 = ' ';

ft = 1, st = 0;

energizer = 0;

system("cls");

display();

display\_high\_score();

gotoxy(Y,X);

cout<<'P';

while(running)

{

Sleep(50);

if (GetAsyncKeyState(VK\_LEFT))

{moveleft();}

if (GetAsyncKeyState(VK\_RIGHT))

{moveright();}

if (GetAsyncKeyState(VK\_UP))

{moveup();}

if (GetAsyncKeyState(VK\_DOWN))

{movedown();}

if (GetAsyncKeyState(VK\_ESCAPE))

{game\_esc();}

if((X==0 && Y==2) ||(X==0 && Y==3)||(X==0 && Y==4))

{

gameend();

game\_end\_score();

store\_score\_array();

}

if(energizer)

{

energycount++;

if(energycount > 100)

{

energizer=0;

energycount=0;

}

}

gotoxy(72,18);

SetConsoleTextAttribute(color,5);

if(stars <=3)

cout<<"Checkpoint Left:"<<3-stars;

else

{cout<<"Checkpoint Left:0";}

SetConsoleTextAttribute(color,10);

if(score > highest\_score)

{SetConsoleTextAttribute(color,10);}

else

{SetConsoleTextAttribute(color, 4);}

gotoxy(77, 5);

score1();

SetConsoleTextAttribute(color,15);

gotoxy(77, 11);

if(lifes == 3)

{

SetConsoleTextAttribute(color,2);

}

if(lifes == 2)

{

SetConsoleTextAttribute(color,6);

}

if(lifes == 1)

{

SetConsoleTextAttribute(color,4);

}

cout<<"Lifes:"<<lifes;

SetConsoleTextAttribute(color,15);

gotoxy(75, 13);

cout<<" ";

gotoxy(75, 13);

if(health==100)

{

SetConsoleTextAttribute(color,1);

cout<<"Health : 100%";

SetConsoleTextAttribute(color,15);

}

if(health==75)

{

SetConsoleTextAttribute(color,11);

cout<<"Health : 75%";

SetConsoleTextAttribute(color,15);

}

if(health==50)

{

SetConsoleTextAttribute(color,14);

cout<<"Health : 50%";

SetConsoleTextAttribute(color,15);

}

if(health==25)

{

SetConsoleTextAttribute(color,4);

cout<<"Health : 25%";

SetConsoleTextAttribute(color,15);

}

if(energizer)

{

gotoxy(75, 20);

cout<<" ";

gotoxy(75, 20);

if(energycount<80)

{SetConsoleTextAttribute(color,6);}

else

{SetConsoleTextAttribute(color,4);}

cout << "Time Left:" << 100 - energycount;

SetConsoleTextAttribute(color,15);

}

if (!energizer)

{

gotoxy(75, 20);

cout << " ";

enemy1();

enemy2();

enemy3();

enemy4();

enemy5();

fire1();

fire2();

fire3();

}

}

}

else if(option=='2')

{

header();

SetConsoleTextAttribute(color,64);

cout<<endl<<endl<<"View Records>"<<endl<<endl;

SetConsoleTextAttribute(color,15);

viewrecords();

}

else if(option == '3')

{

instructions();

}

else if(option=='4')

{

exitgame();

store\_score\_file();

}

else

{

cout<<"You have entered the wrong input"<<endl<<endl<<"Press any key to enter again...";

getch();

}

}

}