#include "graphics.h" ***//To use the inbuilt graphics function***

#include "Board.h" ***//Externally built header file***

#include <iostream> ***//For input-output related functions***

#include <thread> ***//To enable multiprocessing***

#include <streambuf> ***//To use the stream functions***

#include <time.h>  ***//To use the clock functions***

using namespace std;

Board Brd1;

**//-------------------------------------Contains the timer related functions----------------------------------//**

class Timer {

private:

bool Resetted;

bool running;

bool over;

unsigned long beg;

unsigned long end;

public:

int avail\_time;

Timer();

void Start\_Timer(); ***//Starts the timer***

void Stop\_Timer(); ***//Stops the timer***

void Reset\_Timer(); ***//Resets the timer***

bool isRunning(); ***//Checks whether timer is running***

unsigned long Elap\_Time(); ***//Returns the elapsed time***

bool isOver(); ***//Checks whether time is over***

friend void Show\_Watch(); ***//Maintains the timer***

}Tym;

**//=============================END OF CLASS: Timer======================//**

**//----------------------------------*Contains the functions used in the game*--------------------------------//**

class Fnctns\_Usd{

public:

void Wlcm\_scr(); ***//Presents the welcome screen***

void Game\_Scr(); ***//Manages the game screen***

void Inst\_N\_Rules(); ***//Presents the instructions & rules***

void Prsnt\_Menu(); ***//Presents the Main Menu***

void Exit\_Game(); ***//Exits the Game Screen***

void Restart\_Level(); ***//Restarts the level***

void Pause\_N\_Play(); ***//To pause and play the game***

};

**//=========================*END OF CLASS: Fnctns\_Usd*======================//**

**//---------------------------------*Contains the functions for score calculations*---------------------------//**

class High\_Scores{

public:

High\_Scores(); //Constructor

void calc\_score(); //Calculates the score

void give\_bonus(); //Gives the bonus points

int Total\_Score(); //Calculate Total Score

int max\_score, total\_score, score, bonus;

}Hi\_Scr;

**//=========================*END OF CLASS: High\_Scores*======================//**

**//-----------------------------------------*Constructor for the class High Scores*-----------------------------//**

High\_Scores:: High\_Scores()

{

score= 0;

total\_score= 0; ***//Initializing all with zero***

max\_score= 0;

bonus= 0;

}

**//====================================================================//**

**//---------------------------------------*Function for calculating score*----------------------------------------//**

void High\_Scores:: calc\_score()

{

score+= 50;

give\_bonus();

}

**//====================================================================//**

**//---------------------------------------------*Function to give bonus score*------------------------------------//**

void High\_Scores:: give\_bonus()

{

if(Tym.Elap\_Time()<= (5/6)\*Tym.avail\_time && Tym.Elap\_Time()>= (2/3)\*Tym.avail\_time )

{

bonus= score/5;

total\_score= score + bonus;

}

else if(Tym.Elap\_Time()<= (2/3)\*Tym.avail\_time && Tym.Elap\_Time()>= (1/2)\*Tym.avail\_time)

{

bonus= (2\*score)/5;

total\_score= score + bonus;

}

else

{

bonus= score/2;

total\_score= score + bonus;

}

}

**//====================================================================//**

**//----------------------------------------*Function to calculate total score*------------------------------------//**

int High\_Scores:: Total\_Score()

{

calc\_score();

return total\_score;

}

**//====================================================================//**

**//---------------------------------------*Constructor for the class Timer*--------------------------------------//**

Timer::Timer()

{

Resetted = true;

running = false;

over= true;

beg = 0;

end = 0;

avail\_time=60;

}

**//====================================================================//**

**//-------------------------------------------*Function to Show the watch*---------------------------------------//**

void Show\_Watch()

{

for(;;)

{

if(Tym.isRunning())

{

settextstyle(11,0,0);

bgiout<<"Time: "<<Tym.Elap\_Time()<<" Seconds";

outstreamxy(264,50);

if(Tym.isOver())

{

cleardevice();

Tym.Stop\_Timer();

Tym.Reset\_Timer();

cleardevice();

rectangle(140,165,655,290);

rectangle(155,180,640,275);

settextstyle(9,0,1);

bgiout<<"Ooops Time over.. Better luck next time...";

outstreamxy(170,220);

delay(1000);

cleardevice();

rectangle(140,165,655,290);

rectangle(155,180,640,275);

settextstyle(9,0,1);

bgiout<<"Please wait while exiting....";

outstreamxy(215,190);

delay(1000);

exit(0);

}

}

Sleep(900);

}

}

**//====================================================================//**

**//------------------------------------------*Function to Start the Timer*-----------------------------------------//**

void Timer::Start\_Timer()

{

{

if(!running)

{

if(Resetted)

beg = (unsigned long) clock();

else

beg -= end - (unsigned long) clock();

running = true;

Resetted = false;

}

}

}

**//====================================================================//**

**//--------------------------------------------*Function to Stop the Timer*---------------------------------------//**

void Timer::Stop\_Timer()

{

if(running)

{

end = (unsigned long) clock();

running = false;

}

}

**//====================================================================//**

**//--------------------------------------------*Function to Reset the Timer*--------------------------------------//**

void Timer::Reset\_Timer()

{

bool wereRunning = running;

if(wereRunning)

Stop\_Timer();

Resetted = true;

beg = 0;

end = 0;

if(wereRunning)

Start\_Timer();

}

**//====================================================================//**

**//---------------------------------*Function to check whether timer is running*----------------------------//**

bool Timer::isRunning()

{

return running;

}

**//====================================================================//**

**//----------------------------------------*Function to calculate elapsed time*---------------------------------//**

unsigned long Timer::Elap\_Time()

{

if(running)

return ((unsigned long) clock() - beg) / CLOCKS\_PER\_SEC;

else

return (end - beg)/CLOCKS\_PER\_SEC;

}

**//====================================================================//**

**//--------------------------------------*Function to check whether time is over*-----------------------------//**

bool Timer::isOver()

{

if(Tym.Elap\_Time()>=60)

return over;

else

return false;

}

**//====================================================================//**

**//-------------------------------------------*Function to restart the level*---------------------------------------//**

void Fnctns\_Usd:: Restart\_Level()

{

Tym.Stop\_Timer();

cleardevice();

rectangle(210,165,610,290);

rectangle(225,180,595,275);

settextstyle(9,0,1);

bgiout<<"Hit any key to continue....";

outstreamxy(295,220);

getch();

cleardevice();

Tym.Reset\_Timer();

Game\_Scr();

}

**//====================================================================//**

**//-----------------------------------*Function to Pause and Replay the game*--------------------------------//**

void Fnctns\_Usd:: Pause\_N\_Play()

{

int ch;

Tym.Stop\_Timer();

Label\_2:

ch= getch();

switch(ch)

{

case 49: Restart\_Level();

break;

case 32: Tym.Start\_Timer();

break;

case 51: Exit\_Game();

break;

default: cout<<"\a" ;

goto Label\_2;

break;

}

}

**//====================================================================//**

**//--------------------------------------------*Function to Exit the Game*-----------------------------------------//**

void Fnctns\_Usd:: Exit\_Game()

{

Tym.Stop\_Timer();

Tym.Reset\_Timer();

cleardevice();

Prsnt\_Menu();

}

**//====================================================================//**

**//---------------------------------*Function to Present the Welcome Screen*--------------------------------//**

void Fnctns\_Usd:: Wlcm\_scr() //Welcome Screen

{

int i;

int length;

char str[]= "SLIDE GAME!!";

length= strlen(str); //To store the length of the string i.e., Slide Game!!

int j=0;

for(int i=0;i<29;++i)

{

cleardevice();

settextstyle(1,0,2);

outtextxy(-70+i\*12,85,"WELCOME");

outtextxy(799-i\*12,100,"TO THE");

delay(50);

}

for(i=0; i<=length; i++)

{

j+=28;

settextstyle(10,0,5);

bgiout<<str[i];

outstreamxy(230+j,150);

delay(350); //To delay printing

}

delay(2000);

}

**//====================================================================//**

**//----------------------------------*Function to Present the Game Screen*------------------------------------//**

void Fnctns\_Usd:: Game\_Scr()

{

int inpt; //For accepting the input for exit/restart or pause

Brd1.Initialize\_Board();

Brd1.Scramble\_Board();

Brd1.Refresh\_Board();

Brd1.Display\_Board();

Tym.Start\_Timer();

do{

inpt= getch();

switch(inpt)

{

case 49: Restart\_Level();

break;

case 32: Pause\_N\_Play();

break;

case 87: Brd1.Move\_Up((char) inpt);

break;

case 119: Brd1.Move\_Up((char) inpt);

break;

case 83: Brd1.Move\_Down((char) inpt);

break;

case 115: Brd1.Move\_Down((char) inpt);

break;

case 65: Brd1.Move\_Left((char) inpt);

break;

case 97: Brd1.Move\_Left((char) inpt);

break;

case 68: Brd1.Move\_Right((char) inpt);

break;

case 100: Brd1.Move\_Right((char) inpt);

break;

case 51: Exit\_Game();

break;

default: break;

}

}while(!Brd1.Is\_Board\_Solved());

Tym.Stop\_Timer();

Sleep(1500);

cleardevice();

rectangle(140,165,655,290);

rectangle(155,180,640,275);

settextstyle(9,0,1);

bgiout<<"Congratzz!! You've won the game...";

outstreamxy(215,190);

bgiout<<"You took "<<Tym.Elap\_Time()<<" seconds to complete the game..";

outstreamxy(170,220);

bgiout<<"Your Score is "<<Hi\_Scr.Total\_Score();

outstreamxy(285,250);

Tym.Reset\_Timer();

delay(3000);

Prsnt\_Menu();

}

**//====================================================================//**

**//------------------------------------*Function to Show the Instructions & Rules*---------------------------------//**

void Fnctns\_Usd:: Inst\_N\_Rules()

{

int ch;

cleardevice();

int buf[]={95,35,

705,35,

705,405,

95,405,

95,395,

695,395,

695,45,

105,45,

105,395,

95,395,

95,35

};

fillpoly(11,buf);

rectangle(115,55,685,385);

settextstyle(10,0,1);

outtextxy(290,50,"INSTRUCTIONS AND RULES");

settextstyle(3,0,1);

outtextxy(200,90,"1. Use 'W', 'S', 'A' and 'D' for up, down, left & right.");

outtextxy(200,115,"2. Arrange the puzzle within 1 minute.");

outtextxy(200,140,"3. Bonus points for completing the puzzle before 1 minute.");

outtextxy(220,300,"PRESS ESCAPE TO GO BACK TO THE MAIN MENU");

ch= getch();

if(ch==KEY\_ESC)

{

Prsnt\_Menu();

}

else

{

std::cout<<"\a";

Inst\_N\_Rules();

}

}

**//====================================================================//**

**//------------------------------------*Function to Present the Main Menu*------------------------------------//**

void Fnctns\_Usd:: Prsnt\_Menu()

{

cleardevice();

int buf[]={95,35,

705,35,

705,405,

95,405,

95,395,

695,395,

695,45,

105,45,

105,395,

95,395,

95,35

};

rectangle(115,55,685,385);

int Opt; //To get the option from the user.

fillpoly(11,buf);

settextstyle(10,0,1);

outtextxy(350,50,"MAIN MENU");

settextstyle(9,0,1);

outtextxy(300,140,"1. New Game");

outtextxy(300,170,"2. Instructions & Rules");

outtextxy(300,200,"3. Quit");

outtextxy(235,250,"Select any option to continue.... ");

Opt=getch();

switch(Opt)

{

case 49: Game\_Scr();

break;

case 50: Inst\_N\_Rules();

break;

case 51: exit(0);

break;

default: Prsnt\_Menu();

break;

}

}

**//====================================================================//**