```
// Nim.cpp: implementation of the Nim class.
//
//////////////////////////////////////////////////////////////////////

#include <iostream>
#include <ctime>
#include <cstdlib>
#include <fstream>
#include "Nim.h"

using namespace std;

const int savesize = 10;

// ********************************************************************
// * Definition of the member function: Nim::Nim                      *
// * This is the constructor member function.  It takes as its arguemnts*
// * a char array with the players name.  It then seeds the random     *
// * number generator with the time.  It sets the current players name, *
// * wins, losses, and score in the playerinfo struct.  It then        *
// * dynamcially creates an array to hold the high score information.   *
// * It then opens up the save file gamedata.txt and reads in the high  *
// * scores from the file and stores them in the array.  It also checks *
// * for the existance of the save file.                               *
// ********************************************************************

Nim::Nim(char name[])
{
    srand(time(NULL)); // Seeds random number generator with time

    playerinfo.wins = 0;
    playerinfo.losses = 0;
    playerinfo.score = 0;
    playerinfo.playername = name;

    ptr = new data [savesize]; // Dynamically creates an array of structs for the high score info.

    ifstream infile;
    infile.open("gamedata.txt");

    if (!infile) // Checks for the existance of the save file
    {
        system("cls");
        cout << "\nNo input file was found\n\n";
        cout << "Loading Blank Data......\n\n";

        for (int i = 0; i < savesize; i++)
        {
            ptr[i].playername = "Blank";
            ptr[i].wins = 0;
            ptr[i].losses = 0;
            ptr[i].score = -500;
        }
        system("pause");
    }
    else
    {
        for (int i = 0; i < savesize; i++) // Loops while reading in the high score info
        {
            infile >> ptr[i].playername >> ptr[i].wins >> ptr[i].losses >> ptr[i].score;
        }
    }

    infile.close();
}

// ********************************************************************
// * Definition of the member function: Nim::~Nim                     *
// * This is the deconstructor member function. It writes the info from *
// * the current high score array to the save file gamedata.txt.  It    *
// * overwrites whats currently in the file.                           *
// ********************************************************************

Nim::~Nim()
{
    ofstream outfile;
```

```
    outfile.open("gamedata.txt", ios::out);

    for (int i = 0; i < savesize; i++) // Loops while writing high score info to save file
    {
        outfile << ptr[i].playername << " " << ptr[i].wins << " " << ptr[i].losses << " " << ptr[i].score <<
 endl;
    }

    outfile.close();
}

// ************************************************************************
// * Definition of the member function: Nim::random                       *
// * This is the constructor member function.  It takes as it's           *
// * arguements a reference paramter for the currentvalue, an int for     *
// * the low range and an int for the high range.  It uses these ranges   *
// * to randomly pick a number between low and high and passes it back    *
// * to main through the reference parameter.                             *
// ************************************************************************

void Nim::random(int &currentvalue, int low, int high)
{
    currentvalue = low + (high - low) * rand() * (1.0 / RAND_MAX); // Selects random number between low and
high
}

// ************************************************************************
// * Definition of the member function: Nim::GameMenu                     *
// * This is the constructor member function.  It displays the main menu*
// * used for the Game of Nim.                                            *
// ************************************************************************

void Nim::GameMenu()
{
    cout << "Welcome to The Game of Nim " << playerinfo.playername << "\n\n";
    cout << "Wins = " << playerinfo.wins << " Losses = " << playerinfo.losses << " Score = " << playerinfo.s
core << endl << endl;
    cout << "1. Play Game\n";
    cout << "2. View High Scores\n";
    cout << "3. View Rules\n";
    cout << "4. Game Difficulty\n";
    cout << "5. Quit\n\n";
    cout << "Your Choice Is: ";
}

// ************************************************************************
// * Definition of the member function: Nim::CheckWinner                  *
// * This is the constructor member function.  It takes as its arguemnts*
// * an int for the current value and another int for whos turn it is. 0*
// * for the player, and 1 for the computer.  This function returns a     *
// * bool for true if the current player is a winner.  It does this by    *
// * checking if there is only 1 marble left in the pile after their      *
// * guess is subtracted from the pile.  If you win, it increases your    *
// * score by 100, and if you lose, it subtracts 50 from your score.      *
// ************************************************************************

bool Nim::CheckWinner(int currentvalue, int turn)
{
    bool status; // Returns true if current player is winner

    if (currentvalue == 1) // Used if there is only 1 marble left
    {
        status = false;

        if (turn == 0) // Used it's the players turn
        {
            playerinfo.wins++; // Increases wins
            playerinfo.score = playerinfo.score + 100; // Updates score
        }
        else // Used if it's the computers turn
        {
            playerinfo.losses++; // Increases losses
            playerinfo.score = playerinfo.score - 50; // Updates score
        }
    }
```

```
    else
    {
        status = true; // Sets to true if there is more than 1 marble left
    }

return status;
}

// ********************************************************************
// * Definition of the member function: Nim::HighScores               *
// * This is the constructor member function.  It takes as its arguemnts*
// * an int for the current index and reference parameters for, the name*
// * wins, losses, and score.  This function is used to pull the high   *
// * score information stored in the internal array.                    *
// ********************************************************************

void Nim::HighScores(int index, string &name, int &wins, int &losses, int &score)
{
    name = ptr[index].playername;
    wins = ptr[index].wins;
    losses = ptr[index].losses;
    score = ptr[index].score;
}

// ********************************************************************
// * Definition of the member function: Nim::CheckHighScore             *
// * This is the constructor member function.  It takes as its arguemnts*
// * a reference parameter for the current position in the high score   *
// * array.  First it determines if your score is a high score or not.  *
// * If you do have a high score, it changes a bool to true and finds   *
// * the position where you stand in the high score list.  If you have a*
// * high score, it updates the high score list by dropping off the     *
// * score and shifting all of the other scores down by one until your  *
// * at your position.  It then copies the your scores and name into the*
// * correct position.                                *
// ********************************************************************

bool Nim::CheckHighScore(int &position)
{
    bool status = false; // False means you don't have a high score

    for (int i = 10; i >= 0; i--) // Loops while checking if you have a high score
    {
        if (playerinfo.score > ptr[i].score) // Used if your score is greater than the current tested score
        {
            position = i; // Sets position to the current position in the loop
            status = true; // Set to true because you have a high score
        }
    }

    if (status) // If you have a high score...
    {
        for (i = 9; i > position; i--) // Loops while shifting each players score down one until at the posi
tion where you are.
        {
            // Copies position of all variables to the one above it

            ptr[i].playername = ptr[i - 1].playername;
            ptr[i].losses = ptr[i -1].losses;
            ptr[i].score = ptr[i -1].score;
            ptr[i].wins = ptr[i - 1].wins;
        }

        // Sets your scores and name to the current position at which you have acheived in the high score li
st

        ptr[position].playername = playerinfo.playername;
        ptr[position].losses = playerinfo.losses;
        ptr[position].score = playerinfo.score;
        ptr[position].wins = playerinfo.wins;
    }
return status;
}
```