Jonathan Cerniaz

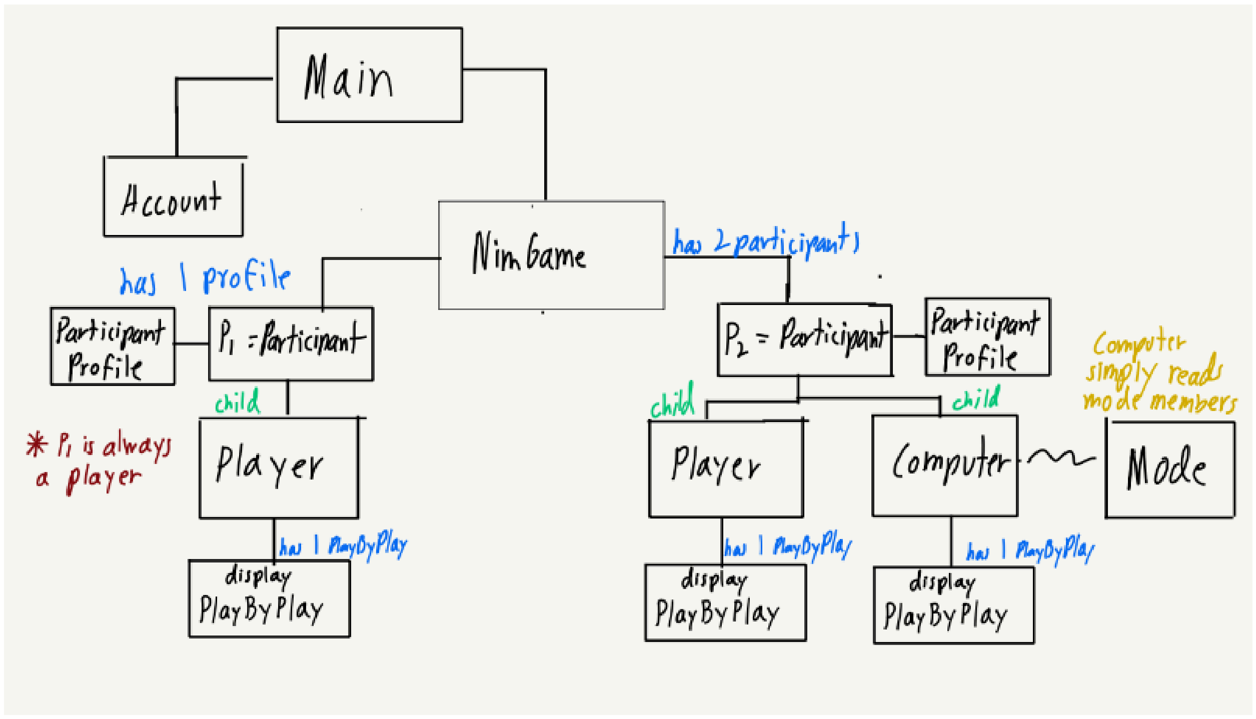
George Elassal

Professor Minthong

11 May 2023

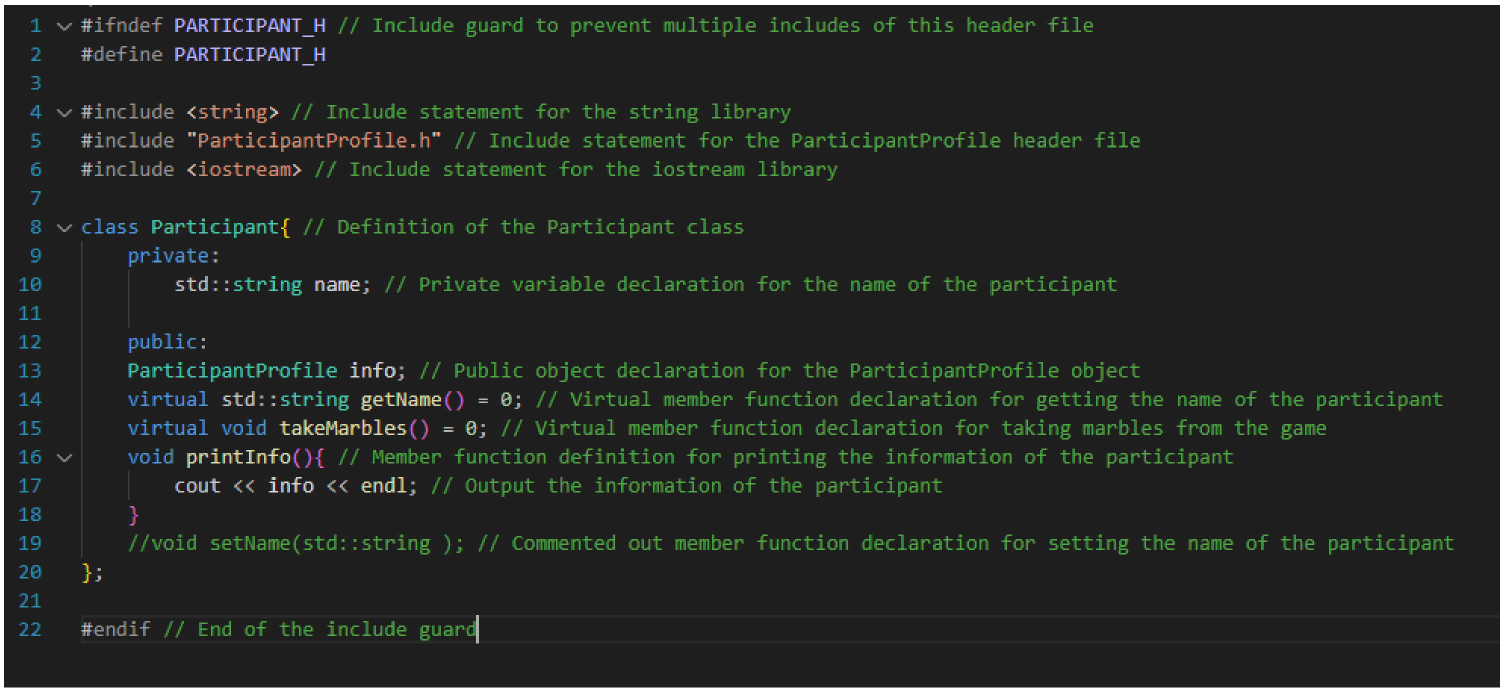
The Game of Nim Report

We will discuss the architecture of this program. First of all, the NimGame class has 4 members. Two static members are integers: order and marbles. They are made to be static so that any other object could read these members without having a NimGame member, and since we never create a new NimGame object more than once, the static members don't create any problems for us. Order can either be a 1 or a 0 which dictates which participant goes first. Marbles is set to be a random number from 10 to 1000, and both these members are randomized everytime a new game has started. It is important to note that playing a new game doesn't create an object of the NimGame class, rather it randomizes the marbles member and order member again. Also, the other two members of NimGame class are participants, named p1 and p2. NimGame class also has a function called "take"; it removes marbles from the pile, or in other words, subtracts an input value from the member "marbles". The Participant class is the parent of the Player and Computer classes. The Participant class also has a member object of the class ParticipantProfile. ParticipantProfile has 9 members which all keep track of the statistics of each participant. The members are **gamesPlayed** (keeps track of games played), **humanGames** (keeps track of games played against a human), **botGames** (keeps track of games played against a computer), **gamesWon** (total games won), **wonHuman** (amount of games won against a human), **wonBot** (amount of games won against a computer), **percentage** (total win percentage), **percenthum** (win percentage against a human), and **percentcomp** (win percentage against a computer). These are updated every time a game has finished, and for player 2 it will keep track of the stats for the local player 2 and the computer player 2. When Displaying stats, it will show you the stats of the player 2 that you just played against (Bot or Human). Now back to the Participant. The Participant class has a member called “name”, which is set in the child classes. Player would request an input for the name, and Computer would automatically set the name to be “Bot”. The Participant has a virtual function called takeMarbles. This is redefined in both the Computer and Player children classes. The Player version requests input from the user to see how many marbles to take and makes sure to validate the input. Before we get into the Computer’s version we must talk about the Mode class. It has one member called level, which is set to 1 or 0 by the user in the main. Now the takeMarbles function in Computer has two different functions depending on the mode member level. If the level is 0, the computer executes the takeMarbles function in the advanced mode. It will take off enough marbles to make the size of the pile a power of two minus one. If it can’t then it just chooses to take off a random number of marbles between 1 and half the pile. If the level is 1, the computer again just chooses to take off a random number of marbles between 1 and half the pile. Now both the Player And Computer class have a member object of the class PlayByPlay called display. The PlayByPlay class has functions that can write to a text file and display what is happening in the game. Now the Account class has a very specific function for us. We chose to mimic other gaming platforms like xbox, steam, and psn. On those platforms, before you play your game you have to sign into your account. So that is how we implemented our Account class. It has two string members, username and password. It also has a static vector of Accounts, that keep track of each account created. So before you can even play the game, you must create an Account and sign in as well. You can create any number of accounts, but you must at least sign into one of them if you want access to the game. This mimics how you would sign into xbox before you can play your games. The main brings all this together. It has menus that show you how to play the game, lets you create an account, sign in, choose to play pvp or pve, display stats, and exit the game.



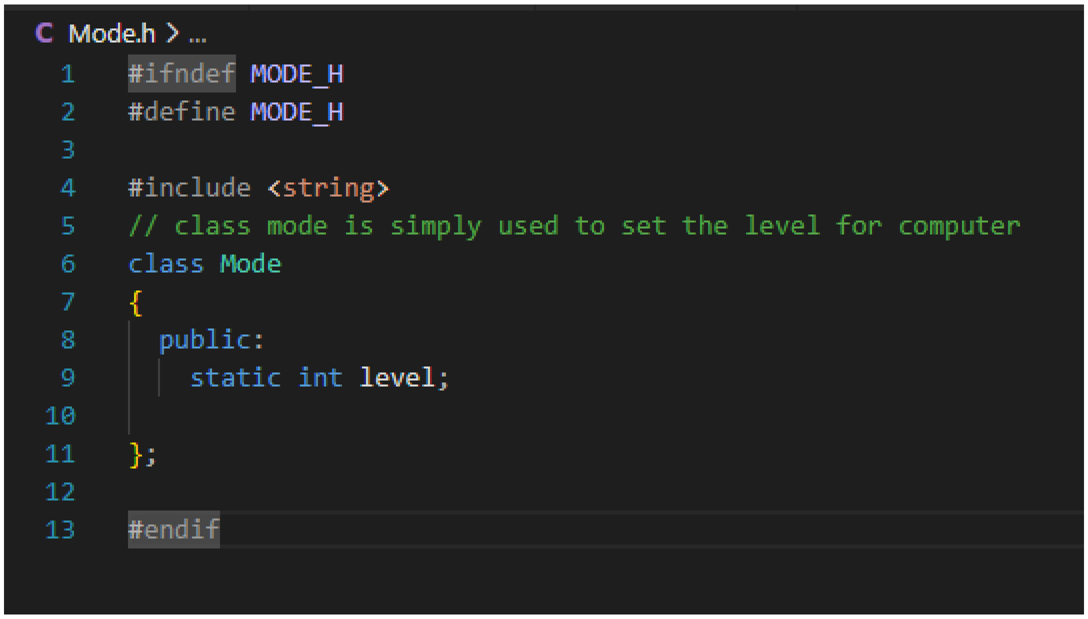
(HTML FILE HAS THE UPDATED COMMENT FORMAT)

Particpant.h:



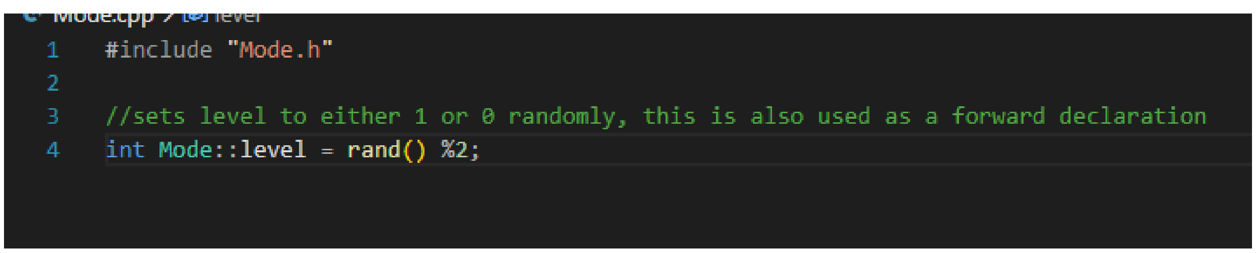
```cpp
#ifndef PARTICIPANT_H // Include guard to prevent multiple includes of this header file
#define PARTICIPANT_H

#include <string> // Include statement for the string library
#include "ParticipantProfile.h" // Include statement for the ParticipantProfile header file
#include <iostream> // Include statement for the iostream library

class Participant{ // Definition of the Participant class
    private:
        std::string name; // Private variable declaration for the name of the participant

    public:
    ParticipantProfile info; // Public object declaration for the ParticipantProfile object
    virtual std::string getName() = 0; // Virtual member function declaration for getting the name of the participant
    virtual void takeMarbles() = 0; // Virtual member function declaration for taking marbles from the game
    void printInfo(){ // Member function definition for printing the information of the participant
        cout << info << endl; // Output the information of the participant
    }
    //void setName(std::string ); // Commented out member function declaration for setting the name of the participant
};

#endif // End of the include guard
```

Mode.h:


Mode.h > ...

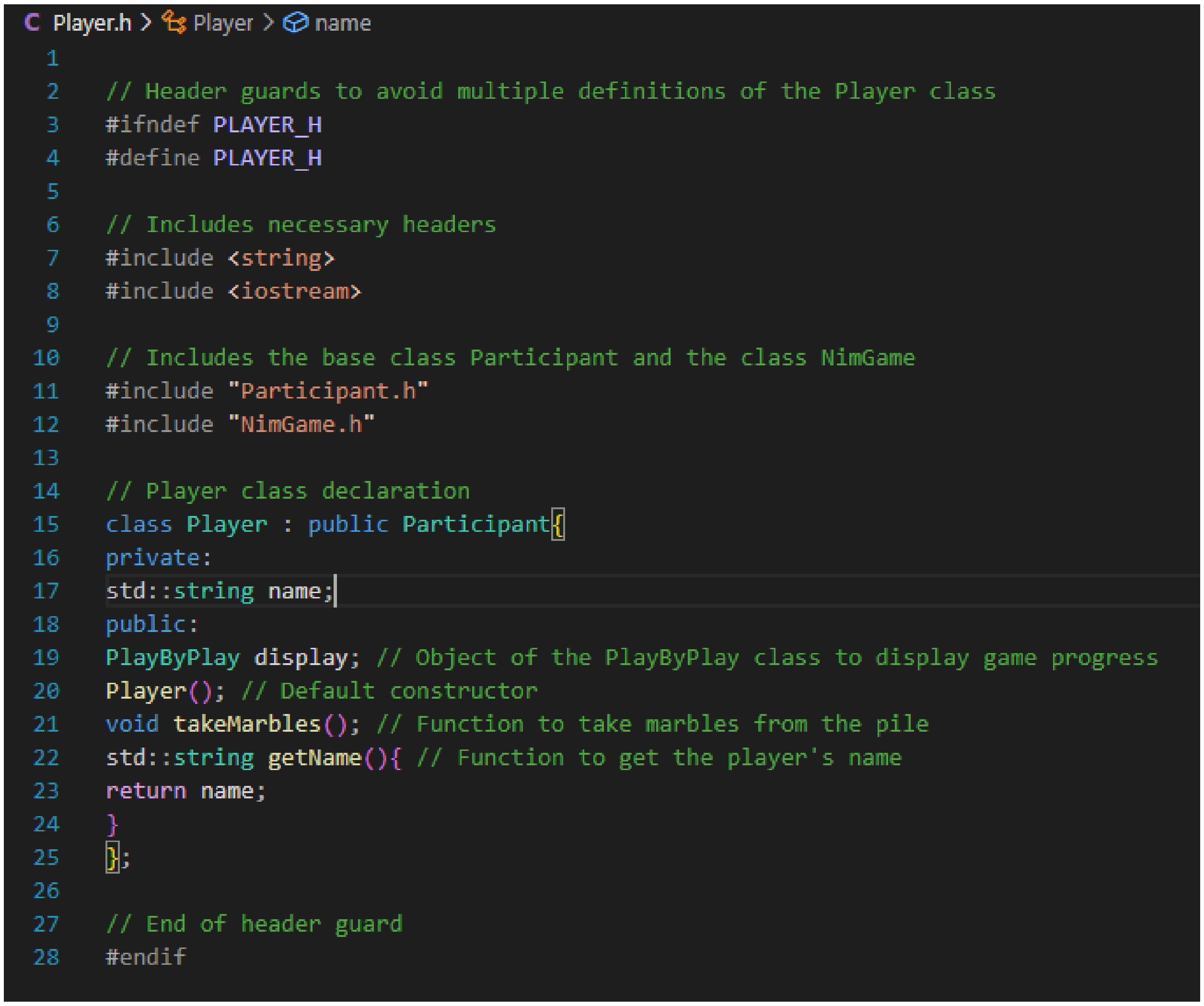
```cpp
#ifndef MODE_H
#define MODE_H

#include <string>
// class mode is simply used to set the level for computer
class Mode
{
  public:
    static int level;

};

#endif
```

Mode.cpp:


Mode.cpp > level

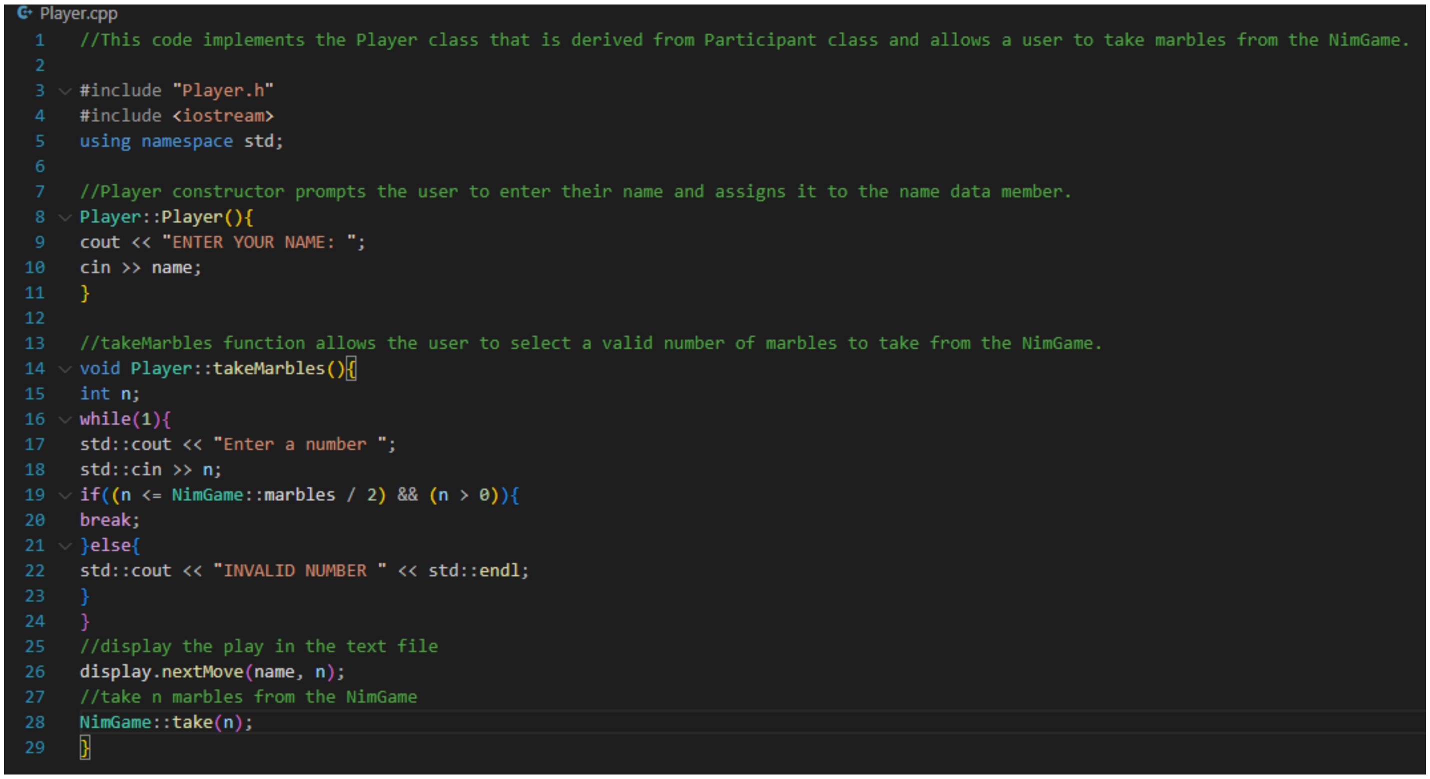
```cpp
#include "Mode.h"

//sets level to either 1 or 0 randomly, this is also used as a forward declaration
int Mode::level = rand() %2;
```

Player.h:



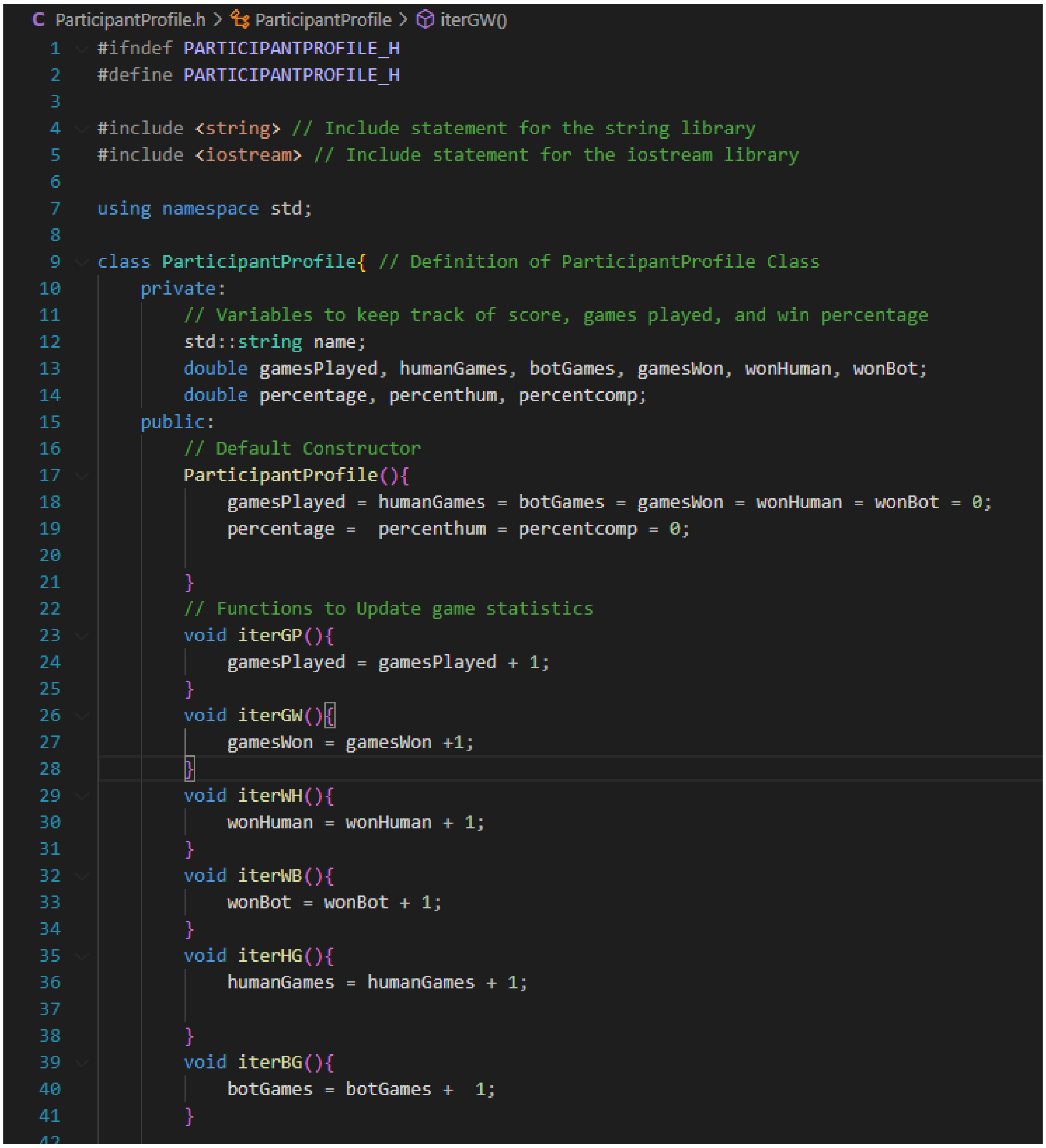
```cpp

// Header guards to avoid multiple definitions of the Player class
#ifndef PLAYER_H
#define PLAYER_H

// Includes necessary headers
#include <string>
#include <iostream>

// Includes the base class Participant and the class NimGame
#include "Participant.h"
#include "NimGame.h"

// Player class declaration
class Player : public Participant{
private:
std::string name;
public:
PlayByPlay display; // Object of the PlayByPlay class to display game progress
Player(); // Default constructor
void takeMarbles(); // Function to take marbles from the pile
std::string getName(){ // Function to get the player's name
return name;
}
};

// End of header guard
#endif
```

Player.cpp:

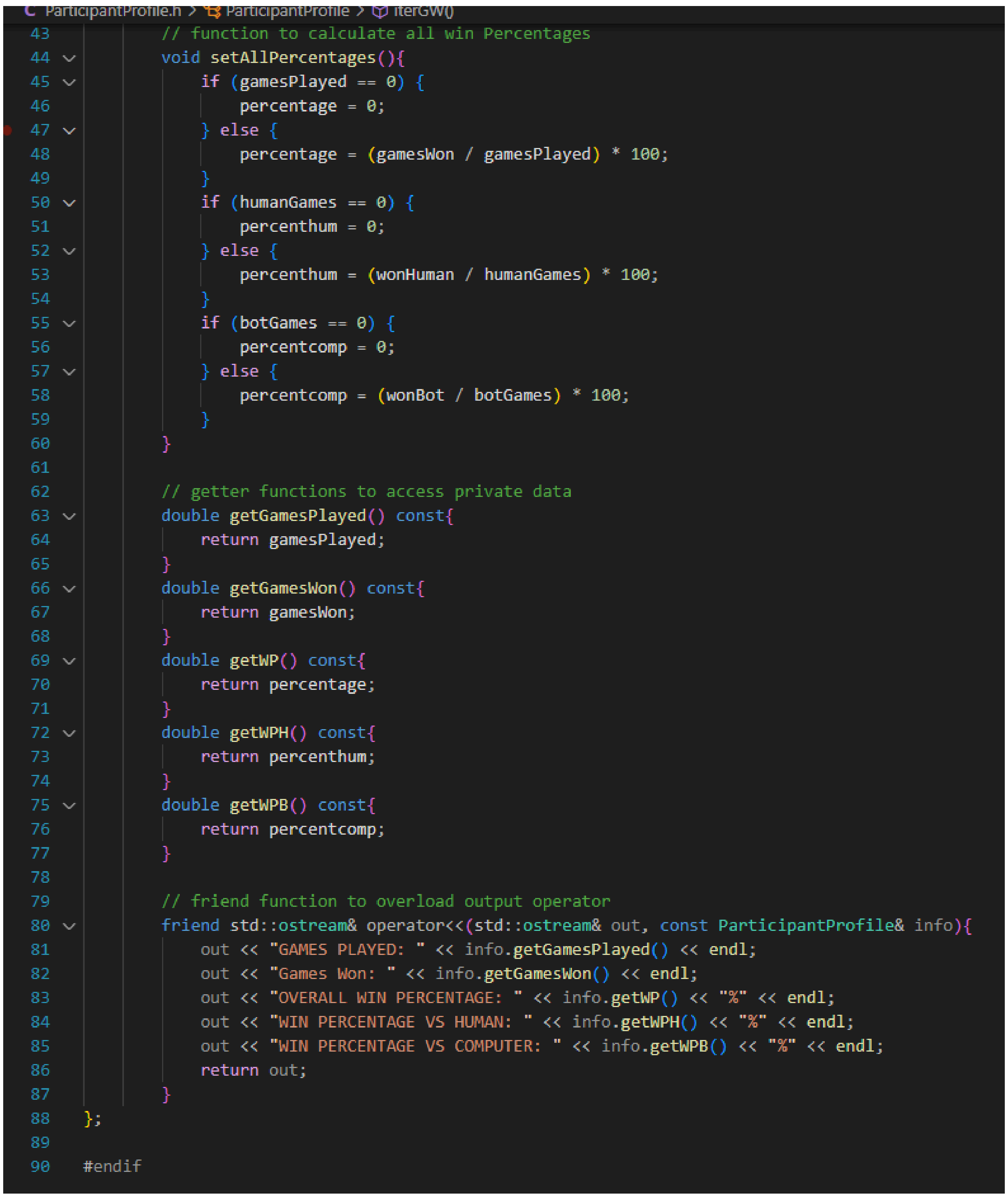

Player.cpp

```cpp
//This code implements the Player class that is derived from Participant class and allows a user to take marbles from the NimGame.

#include "Player.h"
#include <iostream>
using namespace std;

//Player constructor prompts the user to enter their name and assigns it to the name data member.
Player::Player(){
cout << "ENTER YOUR NAME: ";
cin >> name;
}

//takeMarbles function allows the user to select a valid number of marbles to take from the NimGame.
void Player::takeMarbles(){
int n;
while(1){
std::cout << "Enter a number ";
std::cin >> n;
if((n <= NimGame::marbles / 2) && (n > 0)){
break;
}else{
std::cout << "INVALID NUMBER " << std::endl;
}
}
//display the play in the text file
display.nextMove(name, n);
//take n marbles from the NimGame
NimGame::take(n);
}
```

ParticipantProfile.h:

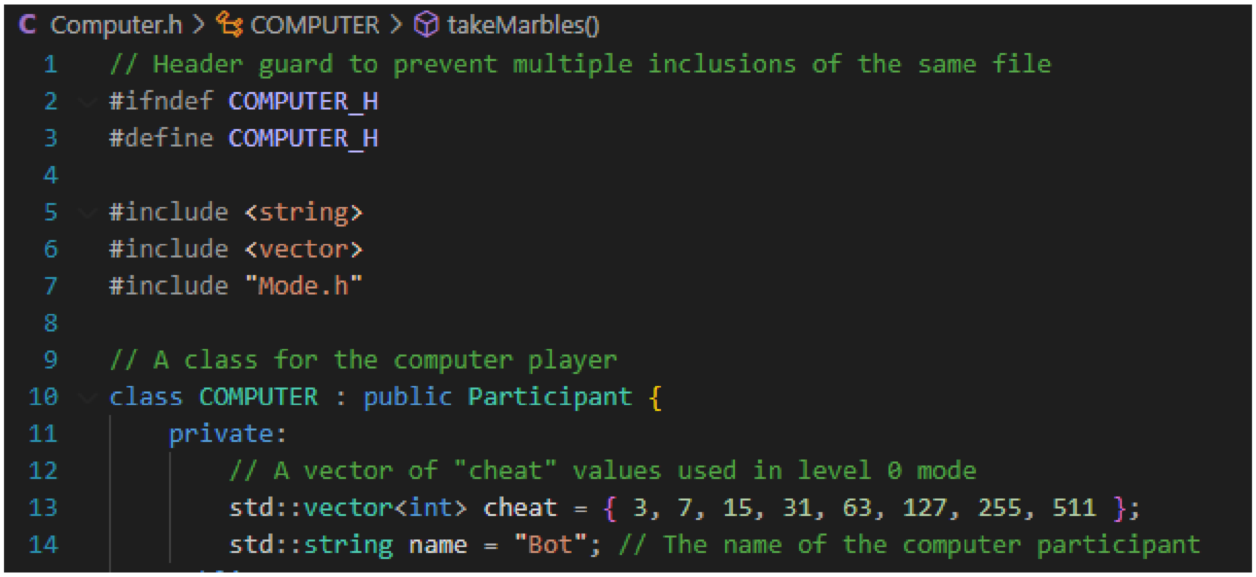


```cpp
#ifndef PARTICIPANTPROFILE_H
#define PARTICIPANTPROFILE_H

#include <string> // Include statement for the string library
#include <iostream> // Include statement for the iostream library

using namespace std;

class ParticipantProfile{ // Definition of ParticipantProfile Class
    private:
        // Variables to keep track of score, games played, and win percentage
        std::string name;
        double gamesPlayed, humanGames, botGames, gamesWon, wonHuman, wonBot;
        double percentage, percenthum, percentcomp;
    public:
        // Default Constructor
        ParticipantProfile(){
            gamesPlayed = humanGames = botGames = gamesWon = wonHuman = wonBot = 0;
            percentage =  percenthum = percentcomp = 0;

        }
        // Functions to Update game statistics
        void iterGP(){
            gamesPlayed = gamesPlayed + 1;
        }
        void iterGW(){
            gamesWon = gamesWon +1;
        }
        void iterWH(){
            wonHuman = wonHuman + 1;
        }
        void iterWB(){
            wonBot = wonBot + 1;
        }
        void iterHG(){
            humanGames = humanGames + 1;

        }
        void iterBG(){
            botGames = botGames +  1;
        }
```

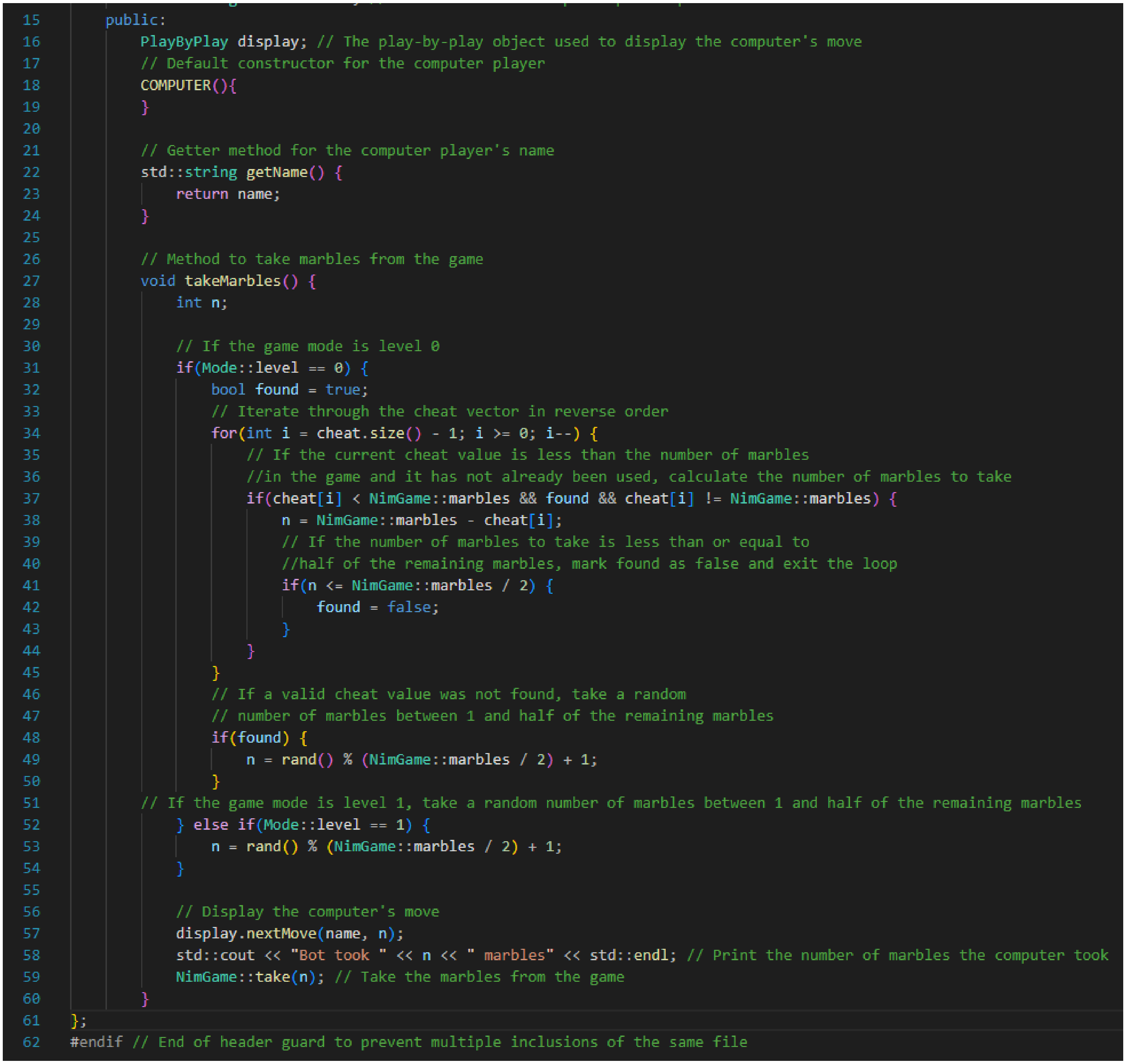

ParticipantProfile.h > ParticipantProfile > iterGW()

```cpp
        // function to calculate all win Percentages
        void setAllPercentages(){
            if (gamesPlayed == 0) {
                percentage = 0;
            } else {
                percentage = (gamesWon / gamesPlayed) * 100;
            }
            if (humanGames == 0) {
                percenthum = 0;
            } else {
                percenthum = (wonHuman / humanGames) * 100;
            }
            if (botGames == 0) {
                percentcomp = 0;
            } else {
                percentcomp = (wonBot / botGames) * 100;
            }
        }

        // getter functions to access private data
        double getGamesPlayed() const{
            return gamesPlayed;
        }
        double getGamesWon() const{
            return gamesWon;
        }
        double getWP() const{
            return percentage;
        }
        double getWPH() const{
            return percenthum;
        }
        double getWPB() const{
            return percentcomp;
        }

        // friend function to overload output operator
        friend std::ostream& operator<<(std::ostream& out, const ParticipantProfile& info){
            out << "GAMES PLAYED: " << info.getGamesPlayed() << endl;
            out << "Games Won: " << info.getGamesWon() << endl;
            out << "OVERALL WIN PERCENTAGE: " << info.getWP() << "%" << endl;
            out << "WIN PERCENTAGE VS HUMAN: " << info.getWPH() << "%" << endl;
            out << "WIN PERCENTAGE VS COMPUTER: " << info.getWPB() << "%" << endl;
            return out;
        }
};

#endif
```

Computer.h:

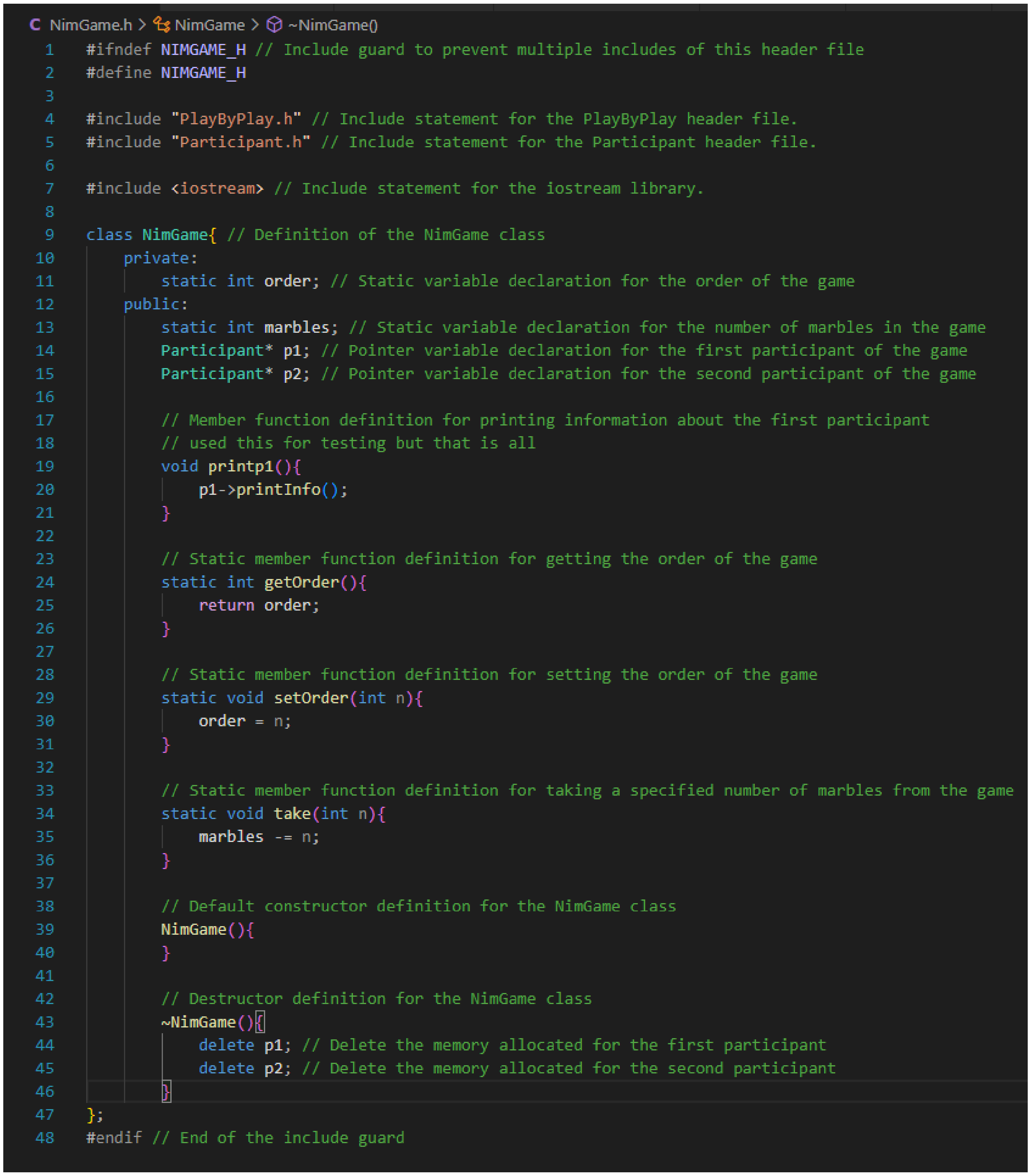

Computer.h > COMPUTER > takeMarbles()

```cpp
// Header guard to prevent multiple inclusions of the same file
#ifndef COMPUTER_H
#define COMPUTER_H

#include <string>
#include <vector>
#include "Mode.h"

// A class for the computer player
class COMPUTER : public Participant {
    private:
        // A vector of "cheat" values used in level 0 mode
        std::vector<int> cheat = { 3, 7, 15, 31, 63, 127, 255, 511 };
        std::string name = "Bot"; // The name of the computer participant
```



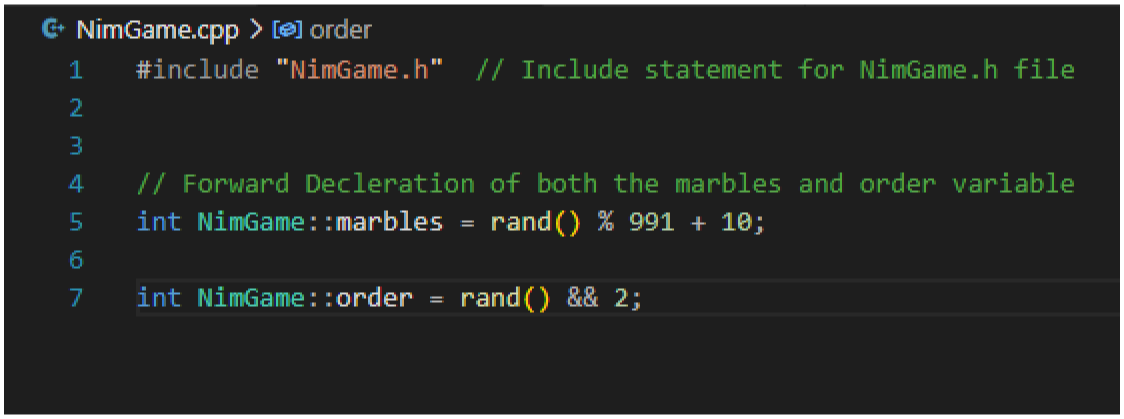
```cpp
    public:
        PlayByPlay display; // The play-by-play object used to display the computer's move
        // Default constructor for the computer player
        COMPUTER(){
        }

        // Getter method for the computer player's name
        std::string getName() {
            return name;
        }

        // Method to take marbles from the game
        void takeMarbles() {
            int n;

            // If the game mode is level 0
            if(Mode::level == 0) {
                bool found = true;
                // Iterate through the cheat vector in reverse order
                for(int i = cheat.size() - 1; i >= 0; i--) {
                    // If the current cheat value is less than the number of marbles
                    //in the game and it has not already been used, calculate the number of marbles to take
                    if(cheat[i] < NimGame::marbles && found && cheat[i] != NimGame::marbles) {
                        n = NimGame::marbles - cheat[i];
                        // If the number of marbles to take is less than or equal to
                        //half of the remaining marbles, mark found as false and exit the loop
                        if(n <= NimGame::marbles / 2) {
                            found = false;
                        }
                    }
                }
                // If a valid cheat value was not found, take a random
                // number of marbles between 1 and half of the remaining marbles
                if(found) {
                    n = rand() % (NimGame::marbles / 2) + 1;
                }
        // If the game mode is level 1, take a random number of marbles between 1 and half of the remaining marbles
            } else if(Mode::level == 1) {
                n = rand() % (NimGame::marbles / 2) + 1;
            }

            // Display the computer's move
            display.nextMove(name, n);
            std::cout << "Bot took " << n << " marbles" << std::endl; // Print the number of marbles the computer took
            NimGame::take(n); // Take the marbles from the game
        }
};
#endif // End of header guard to prevent multiple inclusions of the same file
```

NimGame.h:


NimGame.h > NimGame > ~NimGame()

```cpp
#ifndef NIMGAME_H // Include guard to prevent multiple includes of this header file
#define NIMGAME_H

#include "PlayByPlay.h" // Include statement for the PlayByPlay header file.
#include "Participant.h" // Include statement for the Participant header file.

#include <iostream> // Include statement for the iostream library.

class NimGame{ // Definition of the NimGame class
    private:
        static int order; // Static variable declaration for the order of the game
    public:
        static int marbles; // Static variable declaration for the number of marbles in the game
        Participant* p1; // Pointer variable declaration for the first participant of the game
        Participant* p2; // Pointer variable declaration for the second participant of the game

        // Member function definition for printing information about the first participant
        // used this for testing but that is all
        void printp1(){
            p1->printInfo();
        }

        // Static member function definition for getting the order of the game
        static int getOrder(){
            return order;
        }

        // Static member function definition for setting the order of the game
        static void setOrder(int n){
            order = n;
        }

        // Static member function definition for taking a specified number of marbles from the game
        static void take(int n){
            marbles -= n;
        }

        // Default constructor definition for the NimGame class
        NimGame(){
        }

        // Destructor definition for the NimGame class
        ~NimGame(){
            delete p1; // Delete the memory allocated for the first participant
            delete p2; // Delete the memory allocated for the second participant
        }
};
#endif // End of the include guard
```

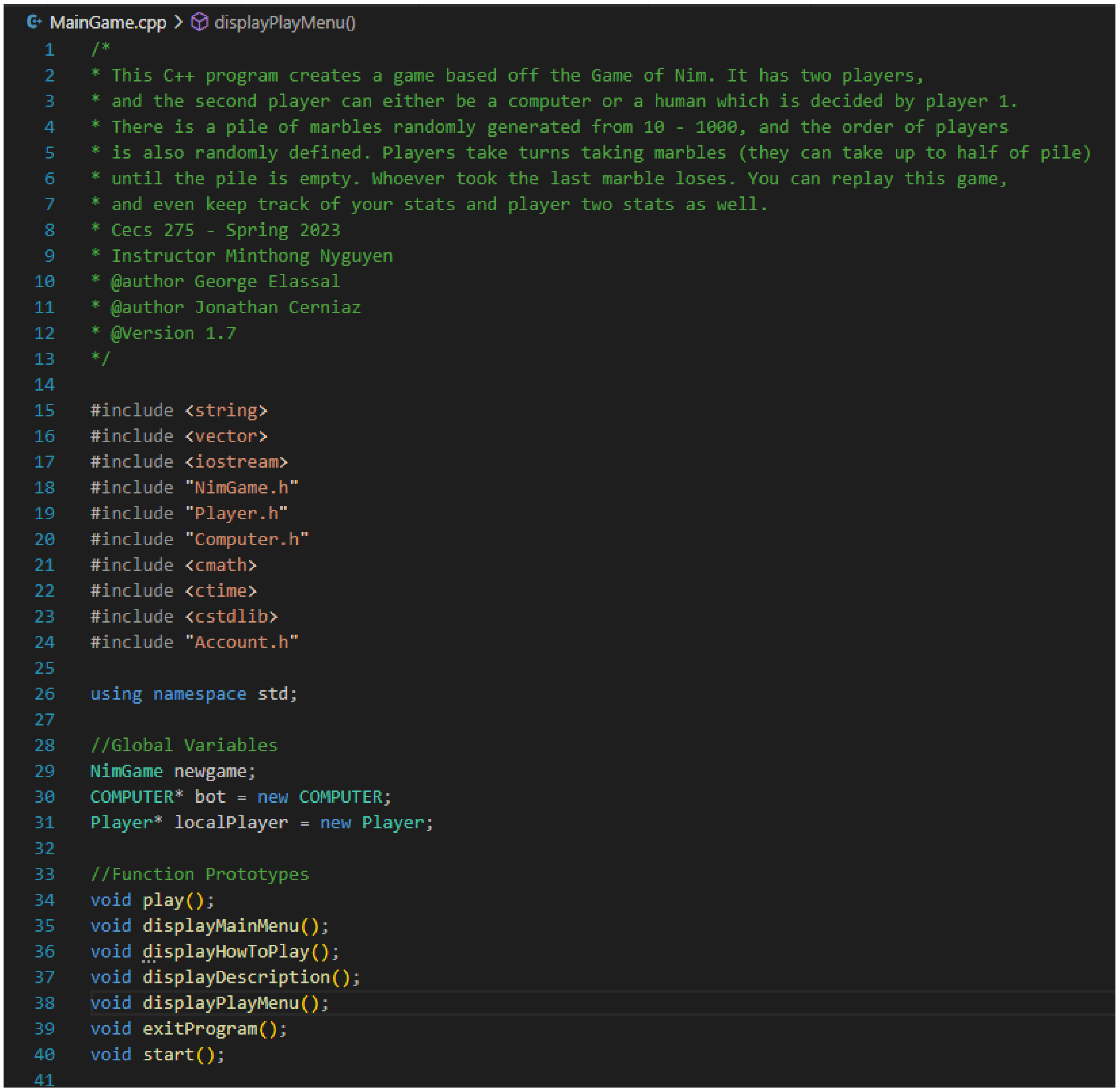
NimGame.cpp



NimGame.cpp > order

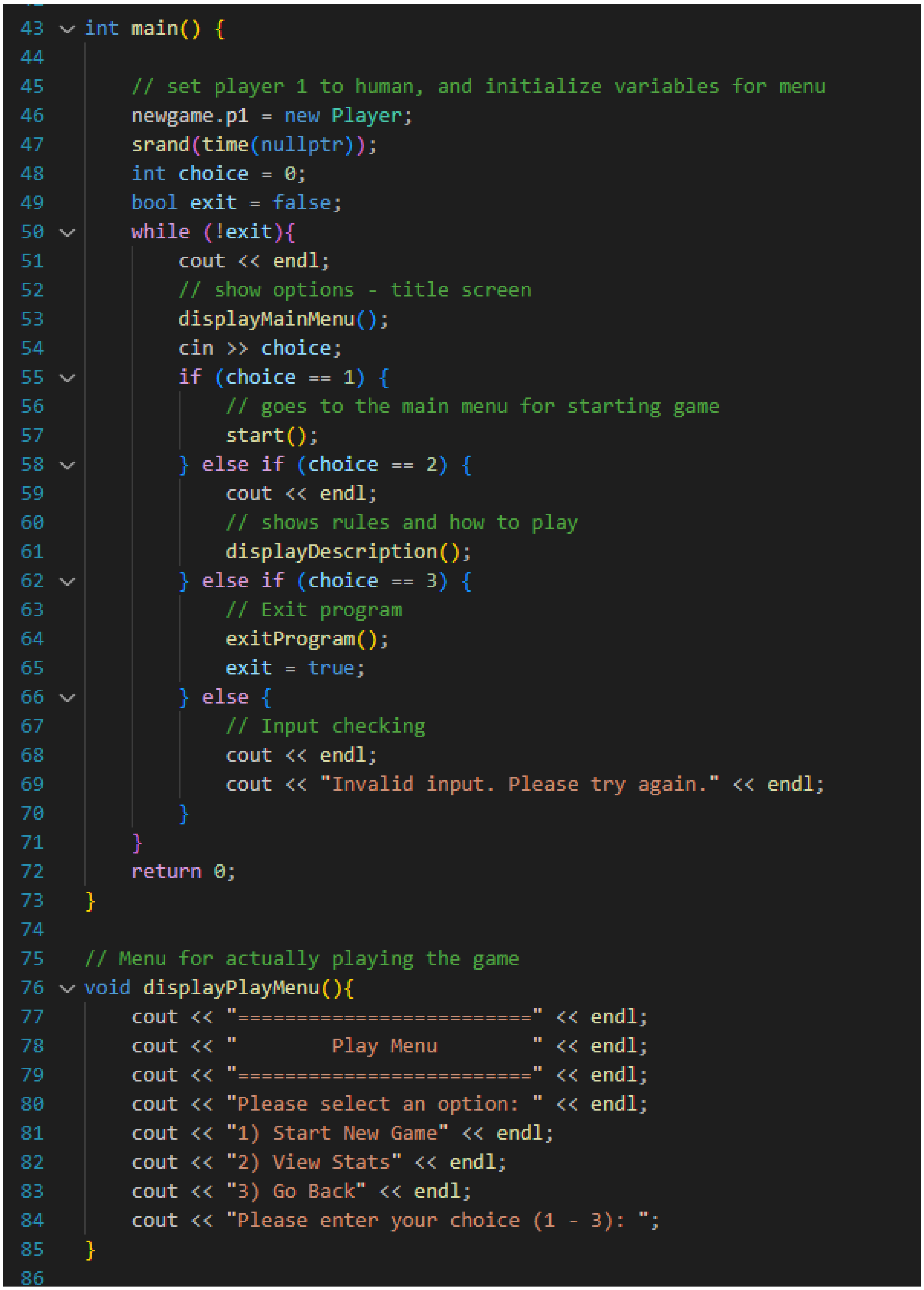
```cpp
#include "NimGame.h"  // Include statement for NimGame.h file


// Forward Decleration of both the marbles and order variable
int NimGame::marbles = rand() % 991 + 10;

int NimGame::order = rand() && 2;
```

MainGame.cpp:

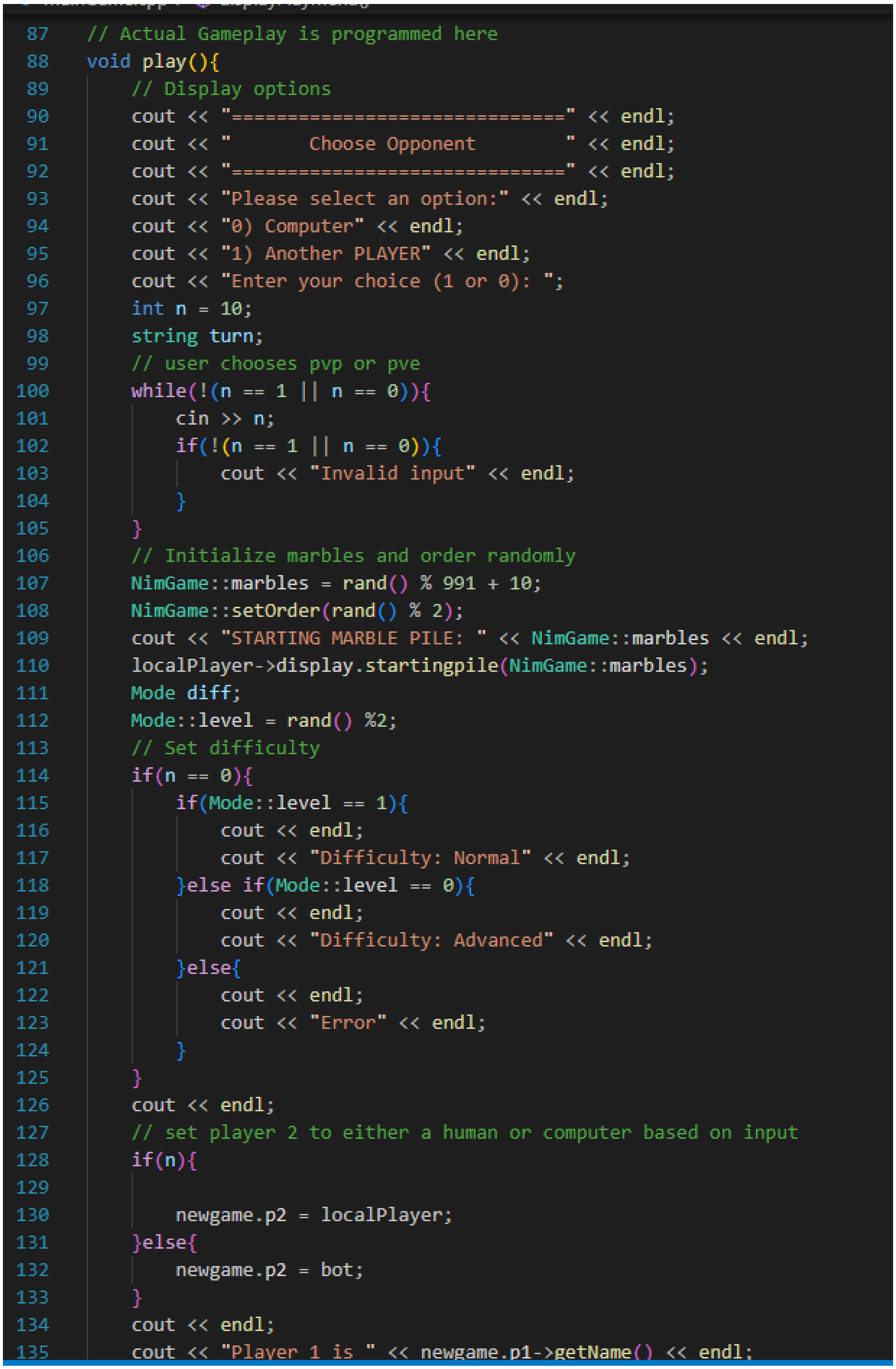


MainGame.cpp > displayPlayMenu()

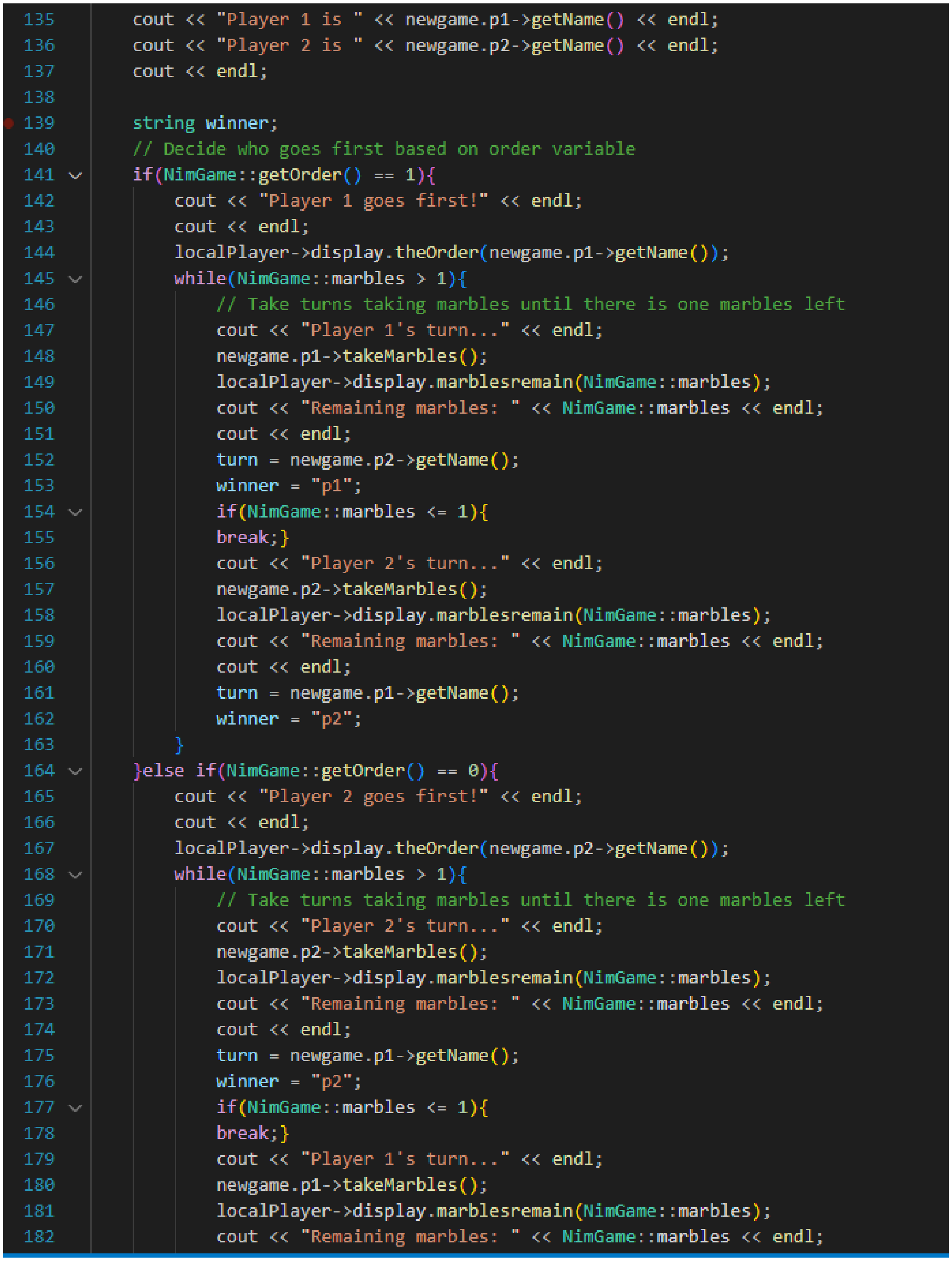
```cpp
/*
* This C++ program creates a game based off the Game of Nim. It has two players,
* and the second player can either be a computer or a human which is decided by player 1.
* There is a pile of marbles randomly generated from 10 - 1000, and the order of players
* is also randomly defined. Players take turns taking marbles (they can take up to half of pile)
* until the pile is empty. Whoever took the last marble loses. You can replay this game,
* and even keep track of your stats and player two stats as well.
* Cecs 275 - Spring 2023
* Instructor Minthong Nyguyen
* @author George Elassal
* @author Jonathan Cerniaz
* @Version 1.7
*/

#include <string>
#include <vector>
#include <iostream>
#include "NimGame.h"
#include "Player.h"
#include "Computer.h"
#include <cmath>
#include <ctime>
#include <cstdlib>
#include "Account.h"

using namespace std;

//Global Variables
NimGame newgame;
COMPUTER* bot = new COMPUTER;
Player* localPlayer = new Player;

//Function Prototypes
void play();
void displayMainMenu();
void displayHowToPlay();
void displayDescription();
void displayPlayMenu();
void exitProgram();
void start();

```



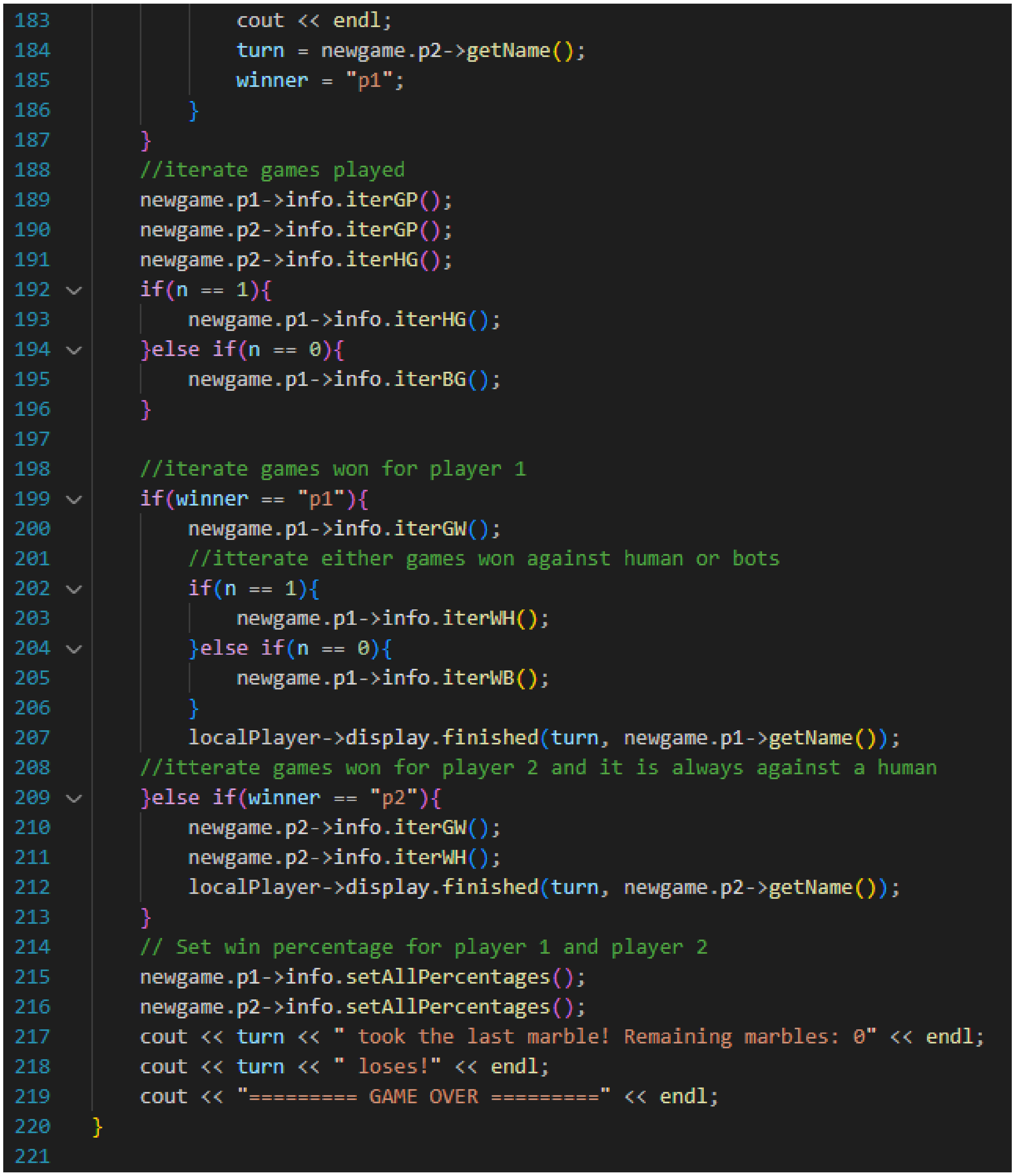
```cpp
int main() {

    // set player 1 to human, and initialize variables for menu
    newgame.p1 = new Player;
    srand(time(nullptr));
    int choice = 0;
    bool exit = false;
    while (!exit){
        cout << endl;
        // show options - title screen
        displayMainMenu();
        cin >> choice;
        if (choice == 1) {
            // goes to the main menu for starting game
            start();
        } else if (choice == 2) {
            cout << endl;
            // shows rules and how to play
            displayDescription();
        } else if (choice == 3) {
            // Exit program
            exitProgram();
            exit = true;
        } else {
            // Input checking
            cout << endl;
            cout << "Invalid input. Please try again." << endl;
        }
    }
    return 0;
}

// Menu for actually playing the game
void displayPlayMenu(){
    cout << "=========================" << endl;
    cout << "         Play Menu        " << endl;
    cout << "=========================" << endl;
    cout << "Please select an option: " << endl;
    cout << "1) Start New Game" << endl;
    cout << "2) View Stats" << endl;
    cout << "3) Go Back" << endl;
    cout << "Please enter your choice (1 - 3): ";
}

```



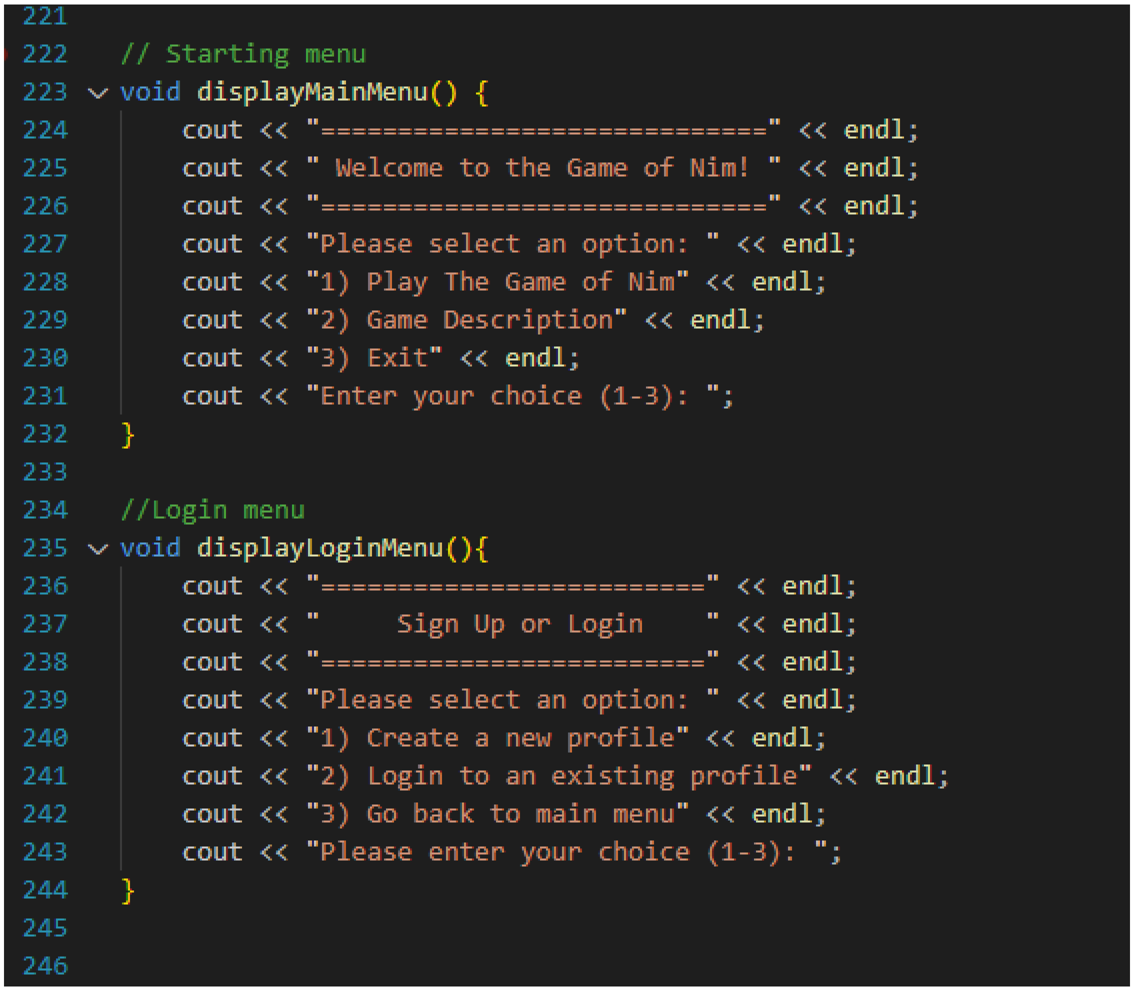
```cpp
// Actual Gameplay is programmed here
void play(){
    // Display options
    cout << "==============================" << endl;
    cout << "       Choose Opponent        " << endl;
    cout << "==============================" << endl;
    cout << "Please select an option:" << endl;
    cout << "0) Computer" << endl;
    cout << "1) Another PLAYER" << endl;
    cout << "Enter your choice (1 or 0): ";
    int n = 10;
    string turn;
    // user chooses pvp or pve
    while(!(n == 1 || n == 0)){
        cin >> n;
        if(!(n == 1 || n == 0)){
            cout << "Invalid input" << endl;
        }
    }
    // Initialize marbles and order randomly
    NimGame::marbles = rand() % 991 + 10;
    NimGame::setOrder(rand() % 2);
    cout << "STARTING MARBLE PILE: " << NimGame::marbles << endl;
    localPlayer->display.startingpile(NimGame::marbles);
    Mode diff;
    Mode::level = rand() %2;
    // Set difficulty
    if(n == 0){
        if(Mode::level == 1){
            cout << endl;
            cout << "Difficulty: Normal" << endl;
        }else if(Mode::level == 0){
            cout << endl;
            cout << "Difficulty: Advanced" << endl;
        }else{
            cout << endl;
            cout << "Error" << endl;
        }
    }
    cout << endl;
    // set player 2 to either a human or computer based on input
    if(n){

        newgame.p2 = localPlayer;
    }else{
        newgame.p2 = bot;
    }
    cout << endl;
    cout << "Player 1 is " << newgame.p1->getName() << endl;
```



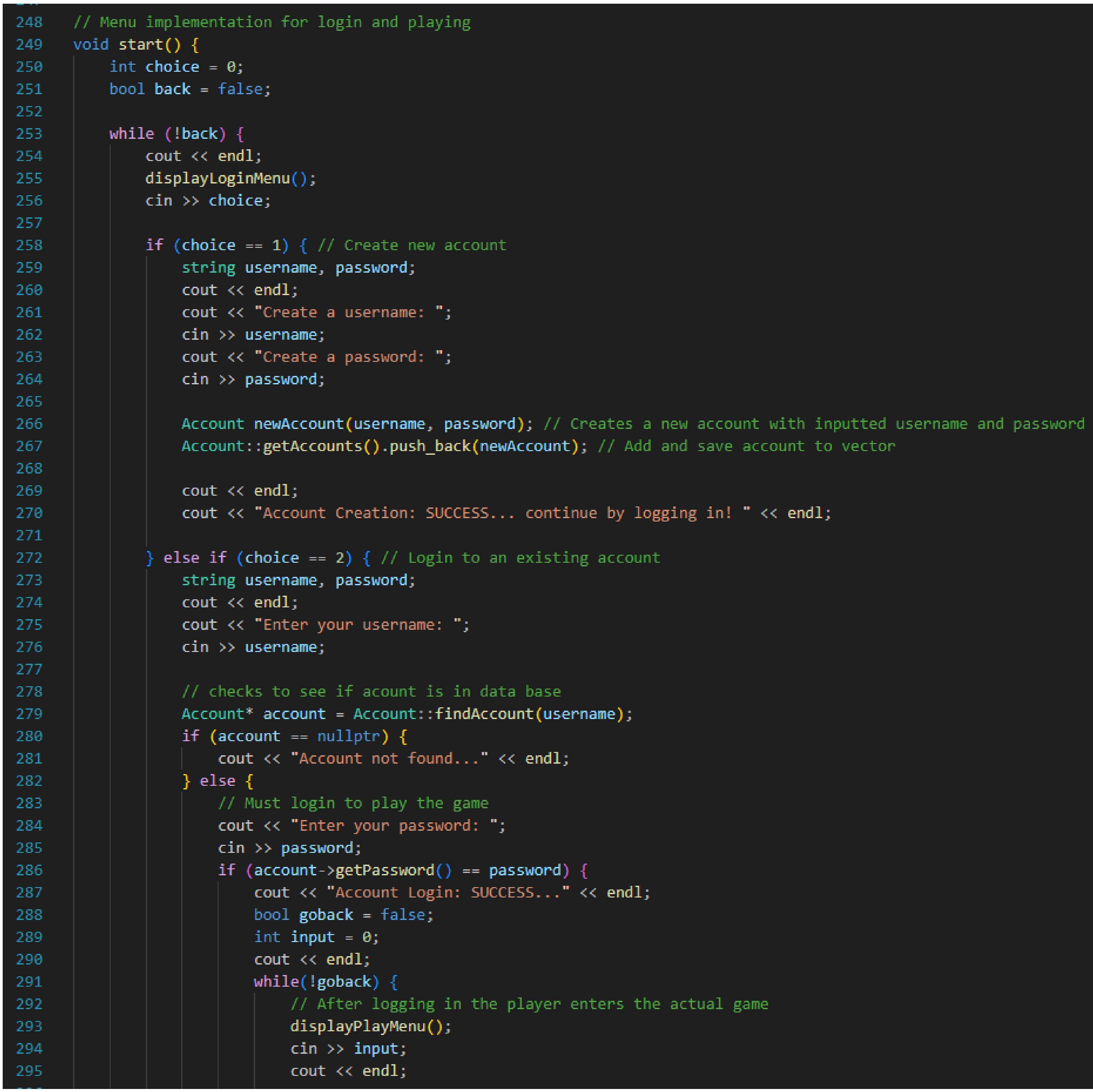
```cpp
    cout << "Player 1 is " << newgame.p1->getName() << endl;
    cout << "Player 2 is " << newgame.p2->getName() << endl;
    cout << endl;

    string winner;
    // Decide who goes first based on order variable
    if(NimGame::getOrder() == 1){
        cout << "Player 1 goes first!" << endl;
        cout << endl;
        localPlayer->display.theOrder(newgame.p1->getName());
        while(NimGame::marbles > 1){
            // Take turns taking marbles until there is one marbles left
            cout << "Player 1's turn..." << endl;
            newgame.p1->takeMarbles();
            localPlayer->display.marblesremain(NimGame::marbles);
            cout << "Remaining marbles: " << NimGame::marbles << endl;
            cout << endl;
            turn = newgame.p2->getName();
            winner = "p1";
            if(NimGame::marbles <= 1){
            break;}
            cout << "Player 2's turn..." << endl;
            newgame.p2->takeMarbles();
            localPlayer->display.marblesremain(NimGame::marbles);
            cout << "Remaining marbles: " << NimGame::marbles << endl;
            cout << endl;
            turn = newgame.p1->getName();
            winner = "p2";
        }
    }else if(NimGame::getOrder() == 0){
        cout << "Player 2 goes first!" << endl;
        cout << endl;
        localPlayer->display.theOrder(newgame.p2->getName());
        while(NimGame::marbles > 1){
            // Take turns taking marbles until there is one marbles left
            cout << "Player 2's turn..." << endl;
            newgame.p2->takeMarbles();
            localPlayer->display.marblesremain(NimGame::marbles);
            cout << "Remaining marbles: " << NimGame::marbles << endl;
            cout << endl;
            turn = newgame.p1->getName();
            winner = "p2";
            if(NimGame::marbles <= 1){
            break;}
            cout << "Player 1's turn..." << endl;
            newgame.p1->takeMarbles();
            localPlayer->display.marblesremain(NimGame::marbles);
            cout << "Remaining marbles: " << NimGame::marbles << endl;
```



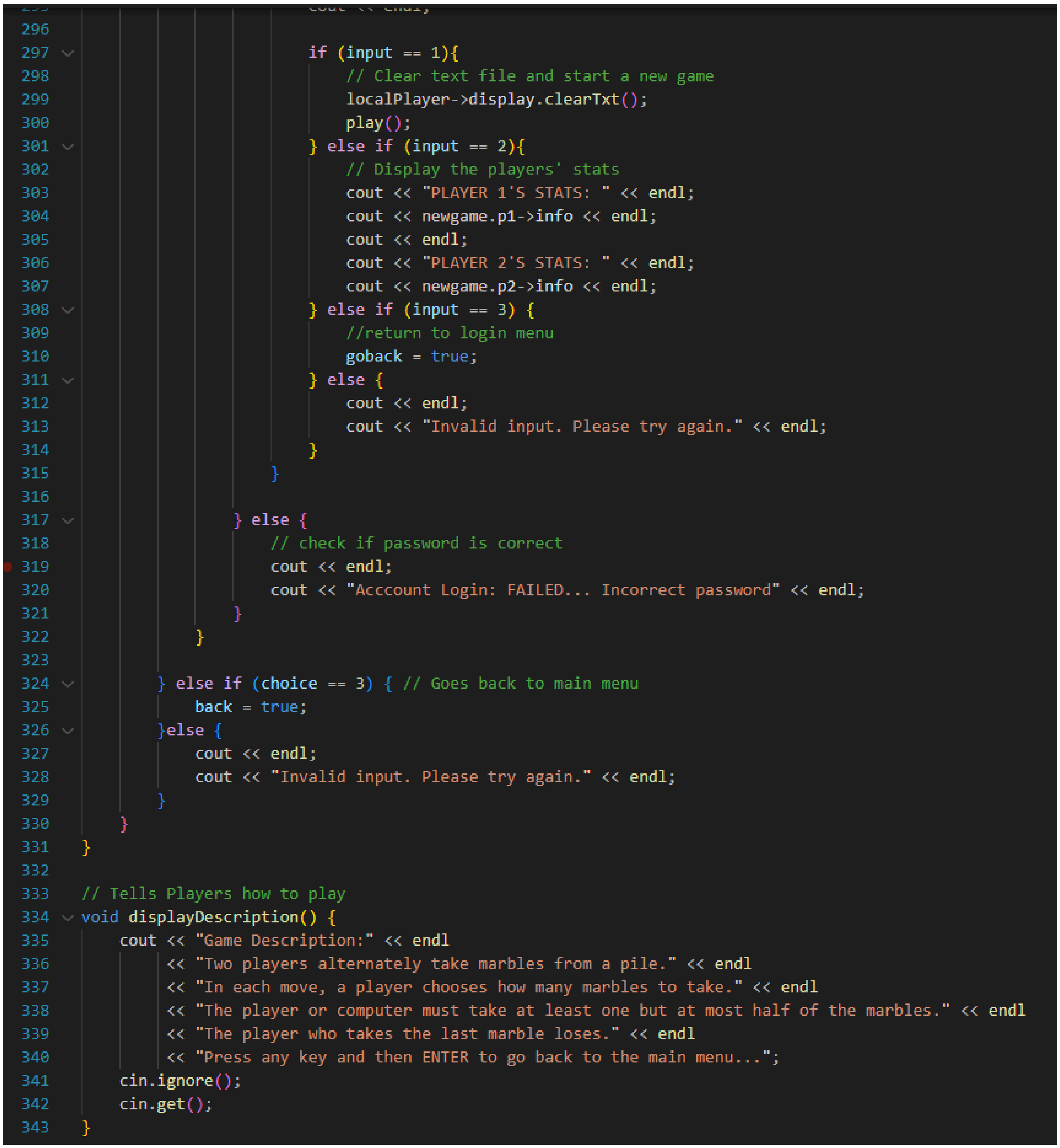
```
            cout << endl;
            turn = newgame.p2->getName();
            winner = "p1";
        }
    }
    //iterate games played
    newgame.p1->info.iterGP();
    newgame.p2->info.iterGP();
    newgame.p2->info.iterHG();
    if(n == 1){
        newgame.p1->info.iterHG();
    }else if(n == 0){
        newgame.p1->info.iterBG();
    }

    //iterate games won for player 1
    if(winner == "p1"){
        newgame.p1->info.iterGW();
        //itterate either games won against human or bots
        if(n == 1){
            newgame.p1->info.iterWH();
        }else if(n == 0){
            newgame.p1->info.iterWB();
        }
        localPlayer->display.finished(turn, newgame.p1->getName());
    //itterate games won for player 2 and it is always against a human
    }else if(winner == "p2"){
        newgame.p2->info.iterGW();
        newgame.p2->info.iterWH();
        localPlayer->display.finished(turn, newgame.p2->getName());
    }
    // Set win percentage for player 1 and player 2
    newgame.p1->info.setAllPercentages();
    newgame.p2->info.setAllPercentages();
    cout << turn << " took the last marble! Remaining marbles: 0" << endl;
    cout << turn << " loses!" << endl;
    cout << "========= GAME OVER =========" << endl;
}

```



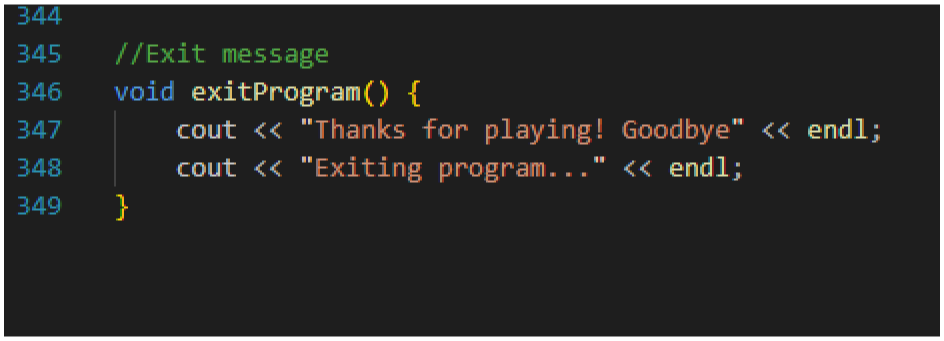
```cpp

// Starting menu
void displayMainMenu() {
    cout << "=============================" << endl;
    cout << " Welcome to the Game of Nim! " << endl;
    cout << "=============================" << endl;
    cout << "Please select an option: " << endl;
    cout << "1) Play The Game of Nim" << endl;
    cout << "2) Game Description" << endl;
    cout << "3) Exit" << endl;
    cout << "Enter your choice (1-3): ";
}

//Login menu
void displayLoginMenu(){
    cout << "=========================" << endl;
    cout << "     Sign Up or Login    " << endl;
    cout << "=========================" << endl;
    cout << "Please select an option: " << endl;
    cout << "1) Create a new profile" << endl;
    cout << "2) Login to an existing profile" << endl;
    cout << "3) Go back to main menu" << endl;
    cout << "Please enter your choice (1-3): ";
}


```



```cpp
// Menu implementation for login and playing
void start() {
    int choice = 0;
    bool back = false;

    while (!back) {
        cout << endl;
        displayLoginMenu();
        cin >> choice;

        if (choice == 1) { // Create new account
            string username, password;
            cout << endl;
            cout << "Create a username: ";
            cin >> username;
            cout << "Create a password: ";
            cin >> password;

            Account newAccount(username, password); // Creates a new account with inputted username and password
            Account::getAccounts().push_back(newAccount); // Add and save account to vector

            cout << endl;
            cout << "Account Creation: SUCCESS... continue by logging in! " << endl;

        } else if (choice == 2) { // Login to an existing account
            string username, password;
            cout << endl;
            cout << "Enter your username: ";
            cin >> username;

            // checks to see if acount is in data base
            Account* account = Account::findAccount(username);
            if (account == nullptr) {
                cout << "Account not found..." << endl;
            } else {
                // Must login to play the game
                cout << "Enter your password: ";
                cin >> password;
                if (account->getPassword() == password) {
                    cout << "Account Login: SUCCESS..." << endl;
                    bool goback = false;
                    int input = 0;
                    cout << endl;
                    while(!goback) {
                        // After logging in the player enters the actual game
                        displayPlayMenu();
                        cin >> input;
                        cout << endl;
```

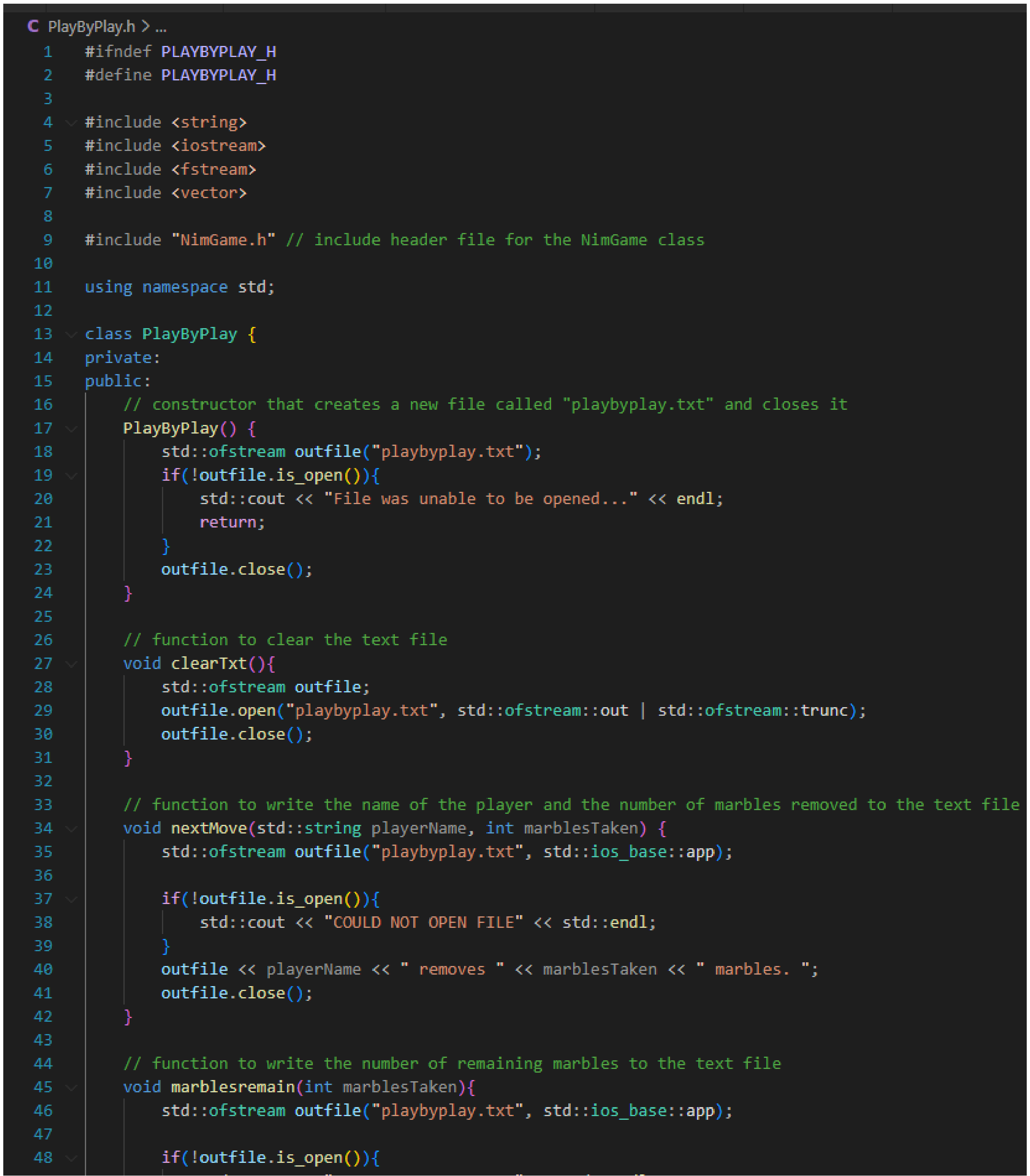


```
                        cout << endl;

                        if (input == 1){
                            // Clear text file and start a new game
                            localPlayer->display.clearTxt();
                            play();
                        } else if (input == 2){
                            // Display the players' stats
                            cout << "PLAYER 1'S STATS: " << endl;
                            cout << newgame.p1->info << endl;
                            cout << endl;
                            cout << "PLAYER 2'S STATS: " << endl;
                            cout << newgame.p2->info << endl;
                        } else if (input == 3) {
                            //return to login menu
                            goback = true;
                        } else {
                            cout << endl;
                            cout << "Invalid input. Please try again." << endl;
                        }
                    }

                } else {
                    // check if password is correct
                    cout << endl;
                    cout << "Acccount Login: FAILED... Incorrect password" << endl;
                }
            }

        } else if (choice == 3) { // Goes back to main menu
            back = true;
        }else {
            cout << endl;
            cout << "Invalid input. Please try again." << endl;
        }
    }
}

// Tells Players how to play
void displayDescription() {
    cout << "Game Description:" << endl
         << "Two players alternately take marbles from a pile." << endl
         << "In each move, a player chooses how many marbles to take." << endl
         << "The player or computer must take at least one but at most half of the marbles." << endl
         << "The player who takes the last marble loses." << endl
         << "Press any key and then ENTER to go back to the main menu...";
    cin.ignore();
    cin.get();
}
```

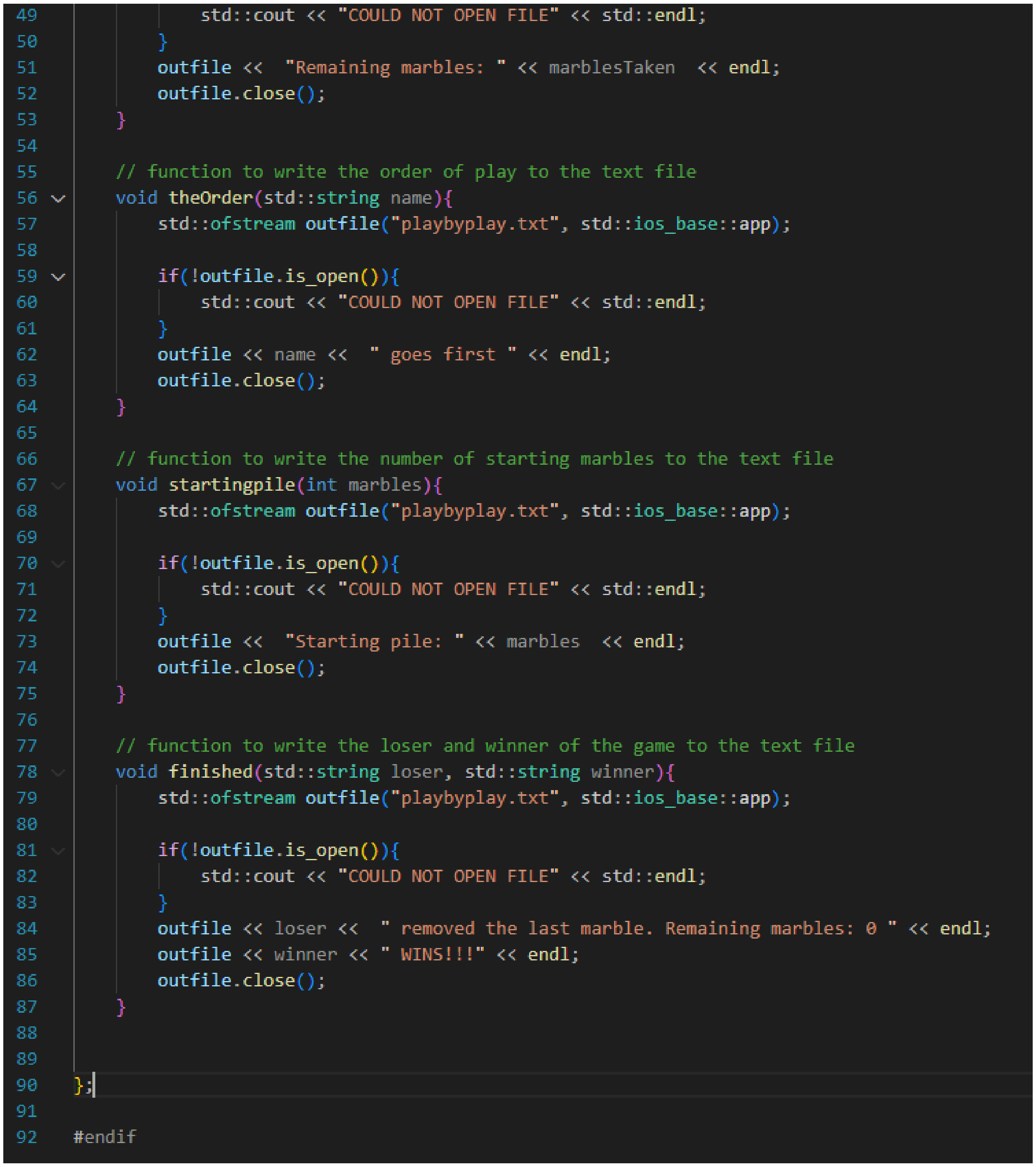


```cpp

//Exit message
void exitProgram() {
    cout << "Thanks for playing! Goodbye" << endl;
    cout << "Exiting program..." << endl;
}
```

PlayByPlay.h:

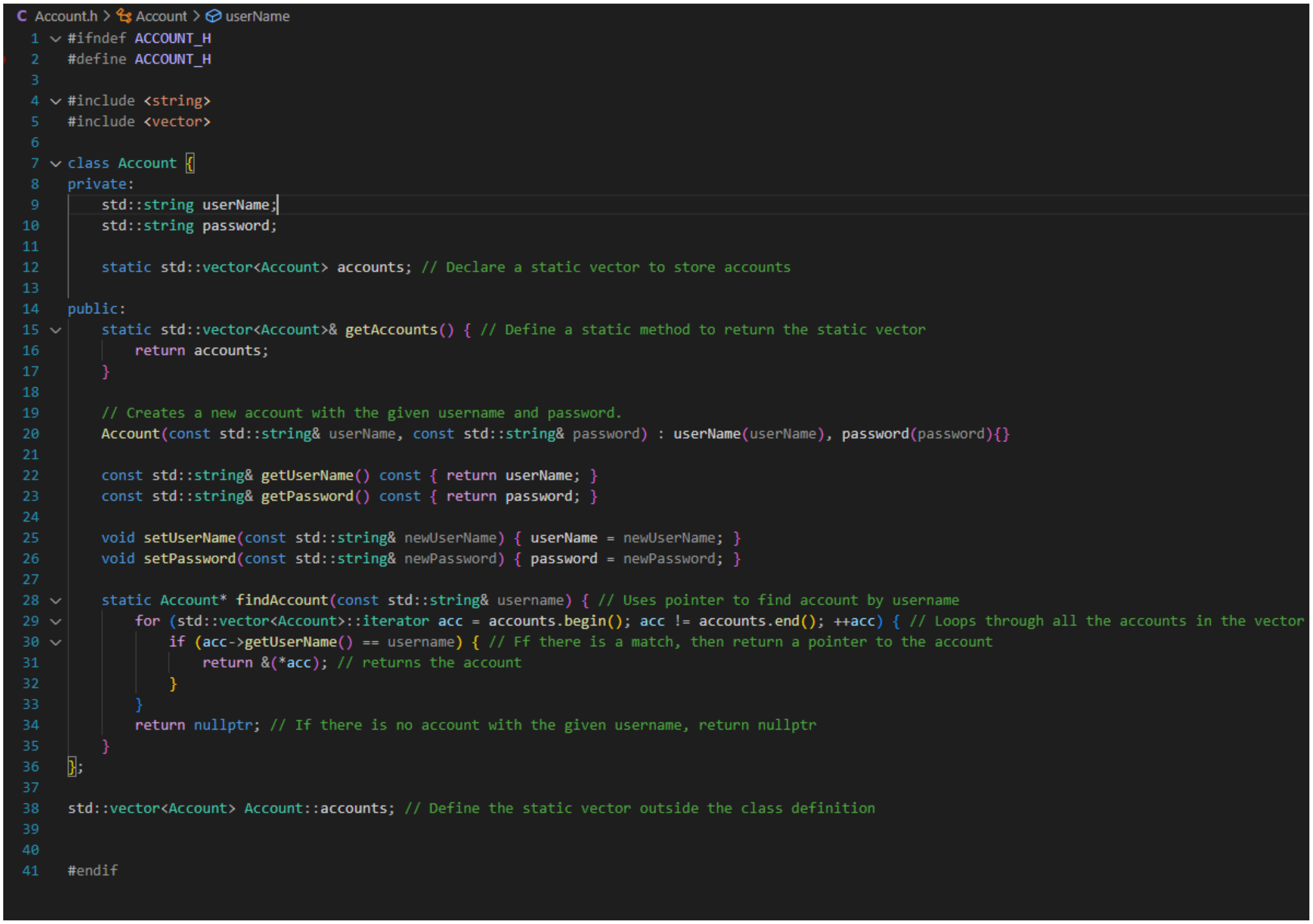


```cpp
#ifndef PLAYBYPLAY_H
#define PLAYBYPLAY_H

#include <string>
#include <iostream>
#include <fstream>
#include <vector>

#include "NimGame.h" // include header file for the NimGame class

using namespace std;

class PlayByPlay {
private:
public:
    // constructor that creates a new file called "playbyplay.txt" and closes it
    PlayByPlay() {
        std::ofstream outfile("playbyplay.txt");
        if(!outfile.is_open()){
            std::cout << "File was unable to be opened..." << endl;
            return;
        }
        outfile.close();
    }

    // function to clear the text file
    void clearTxt(){
        std::ofstream outfile;
        outfile.open("playbyplay.txt", std::ofstream::out | std::ofstream::trunc);
        outfile.close();
    }

    // function to write the name of the player and the number of marbles removed to the text file
    void nextMove(std::string playerName, int marblesTaken) {
        std::ofstream outfile("playbyplay.txt", std::ios_base::app);

        if(!outfile.is_open()){
            std::cout << "COULD NOT OPEN FILE" << std::endl;
        }
        outfile << playerName << " removes " << marblesTaken << " marbles. ";
        outfile.close();
    }

    // function to write the number of remaining marbles to the text file
    void marblesremain(int marblesTaken){
        std::ofstream outfile("playbyplay.txt", std::ios_base::app);

        if(!outfile.is_open()){
```



```cpp
                std::cout << "COULD NOT OPEN FILE" << std::endl;
            }
            outfile <<  "Remaining marbles: " << marblesTaken  << endl;
            outfile.close();
        }

        // function to write the order of play to the text file
        void theOrder(std::string name){
            std::ofstream outfile("playbyplay.txt", std::ios_base::app);

            if(!outfile.is_open()){
                std::cout << "COULD NOT OPEN FILE" << std::endl;
            }
            outfile << name <<  " goes first " << endl;
            outfile.close();
        }

        // function to write the number of starting marbles to the text file
        void startingpile(int marbles){
            std::ofstream outfile("playbyplay.txt", std::ios_base::app);

            if(!outfile.is_open()){
                std::cout << "COULD NOT OPEN FILE" << std::endl;
            }
            outfile <<  "Starting pile: " << marbles  << endl;
            outfile.close();
        }

        // function to write the loser and winner of the game to the text file
        void finished(std::string loser, std::string winner){
            std::ofstream outfile("playbyplay.txt", std::ios_base::app);

            if(!outfile.is_open()){
                std::cout << "COULD NOT OPEN FILE" << std::endl;
            }
            outfile << loser <<  " removed the last marble. Remaining marbles: 0 " << endl;
            outfile << winner << " WINS!!!" << endl;
            outfile.close();
        }


};

#endif
```

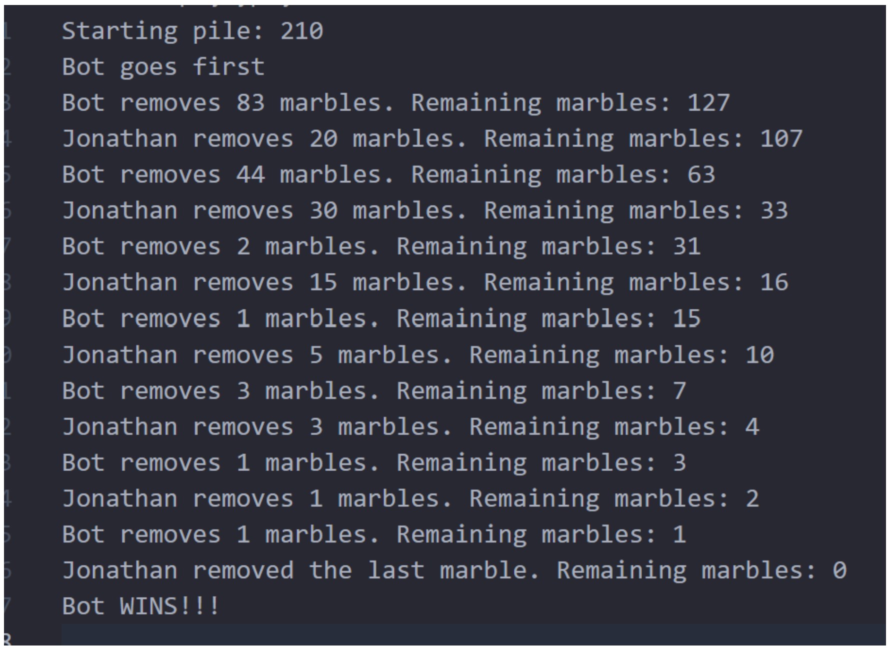
Account.h:



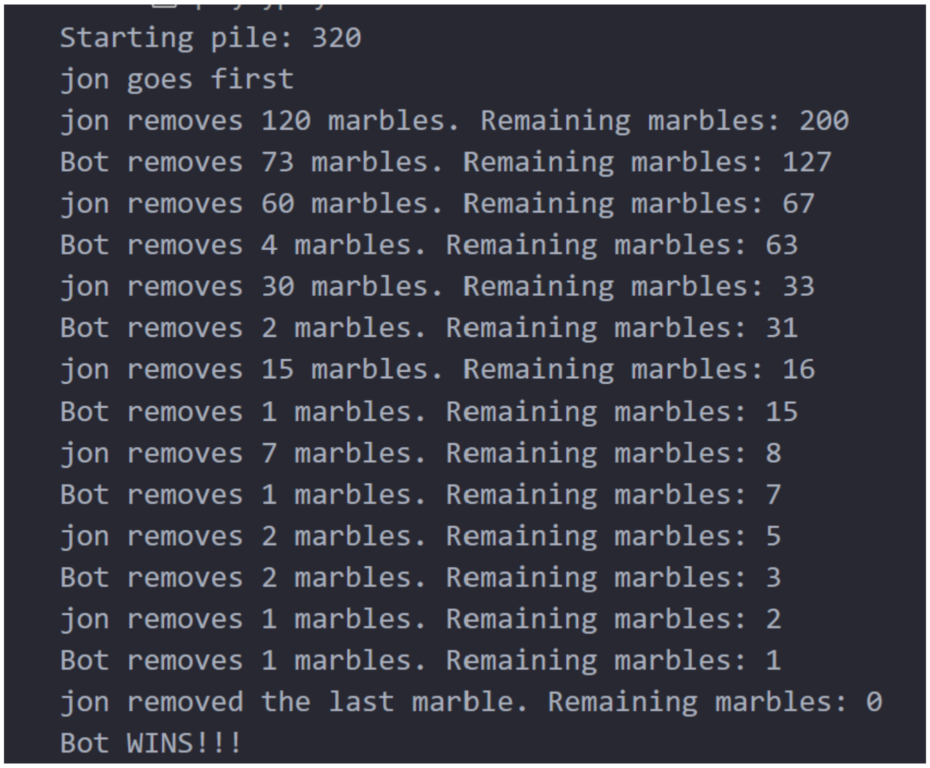
```cpp
#ifndef ACCOUNT_H
#define ACCOUNT_H

#include <string>
#include <vector>

class Account {
private:
    std::string userName;
    std::string password;

    static std::vector<Account> accounts; // Declare a static vector to store accounts

public:
    static std::vector<Account>& getAccounts() { // Define a static method to return the static vector
        return accounts;
    }

    // Creates a new account with the given username and password.
    Account(const std::string& userName, const std::string& password) : userName(userName), password(password){}

    const std::string& getUserName() const { return userName; }
    const std::string& getPassword() const { return password; }

    void setUserName(const std::string& newUserName) { userName = newUserName; }
    void setPassword(const std::string& newPassword) { password = newPassword; }

    static Account* findAccount(const std::string& username) { // Uses pointer to find account by username
        for (std::vector<Account>::iterator acc = accounts.begin(); acc != accounts.end(); ++acc) { // Loops through all the accounts in the vector
            if (acc->getUserName() == username) { // Ff there is a match, then return a pointer to the account
                return &(*acc); // returns the account
            }
        }
        return nullptr; // If there is no account with the given username, return nullptr
    }
};

std::vector<Account> Account::accounts; // Define the static vector outside the class definition


#endif
```

PlayByPlay.txt:

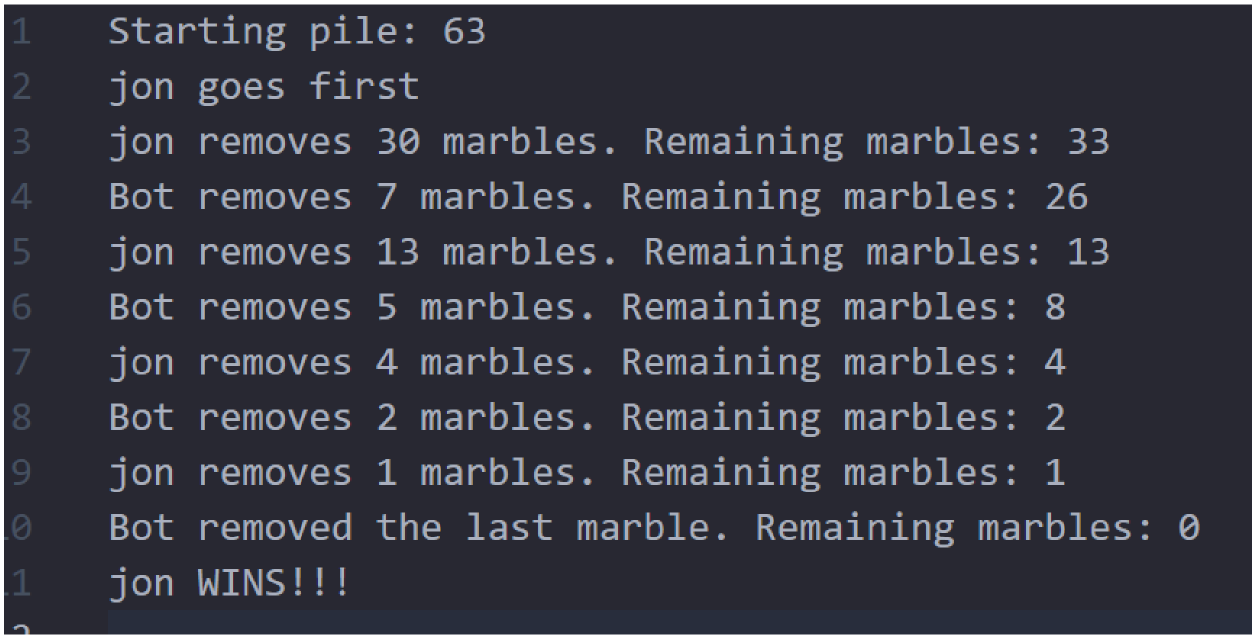
Against Computer:



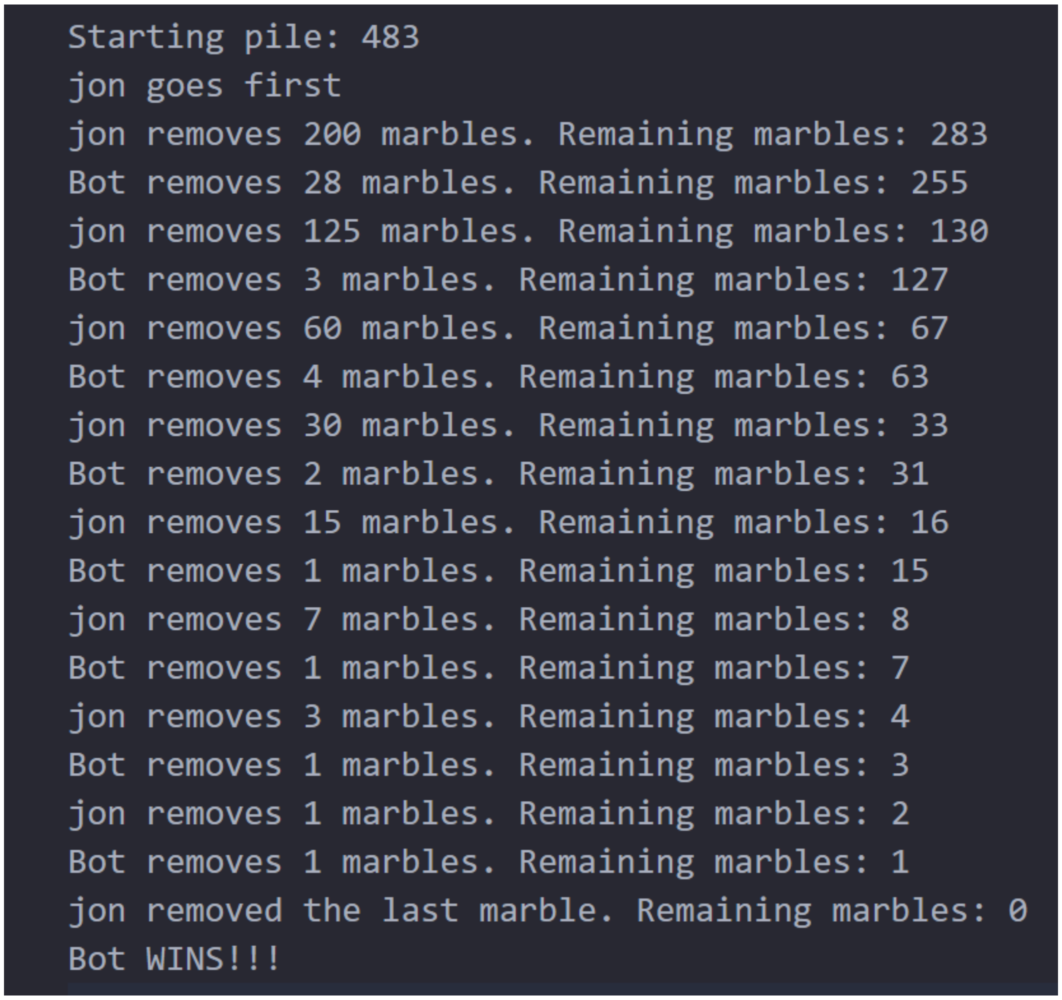
```
Starting pile: 210
Bot goes first
Bot removes 83 marbles. Remaining marbles: 127
Jonathan removes 20 marbles. Remaining marbles: 107
Bot removes 44 marbles. Remaining marbles: 63
Jonathan removes 30 marbles. Remaining marbles: 33
Bot removes 2 marbles. Remaining marbles: 31
Jonathan removes 15 marbles. Remaining marbles: 16
Bot removes 1 marbles. Remaining marbles: 15
Jonathan removes 5 marbles. Remaining marbles: 10
Bot removes 3 marbles. Remaining marbles: 7
Jonathan removes 3 marbles. Remaining marbles: 4
Bot removes 1 marbles. Remaining marbles: 3
Jonathan removes 1 marbles. Remaining marbles: 2
Bot removes 1 marbles. Remaining marbles: 1
Jonathan removed the last marble. Remaining marbles: 0
Bot WINS!!!
```



```
Starting pile: 320
jon goes first
jon removes 120 marbles. Remaining marbles: 200
Bot removes 73 marbles. Remaining marbles: 127
jon removes 60 marbles. Remaining marbles: 67
Bot removes 4 marbles. Remaining marbles: 63
jon removes 30 marbles. Remaining marbles: 33
Bot removes 2 marbles. Remaining marbles: 31
jon removes 15 marbles. Remaining marbles: 16
Bot removes 1 marbles. Remaining marbles: 15
jon removes 7 marbles. Remaining marbles: 8
Bot removes 1 marbles. Remaining marbles: 7
jon removes 2 marbles. Remaining marbles: 5
Bot removes 2 marbles. Remaining marbles: 3
jon removes 1 marbles. Remaining marbles: 2
Bot removes 1 marbles. Remaining marbles: 1
jon removed the last marble. Remaining marbles: 0
Bot WINS!!!
```

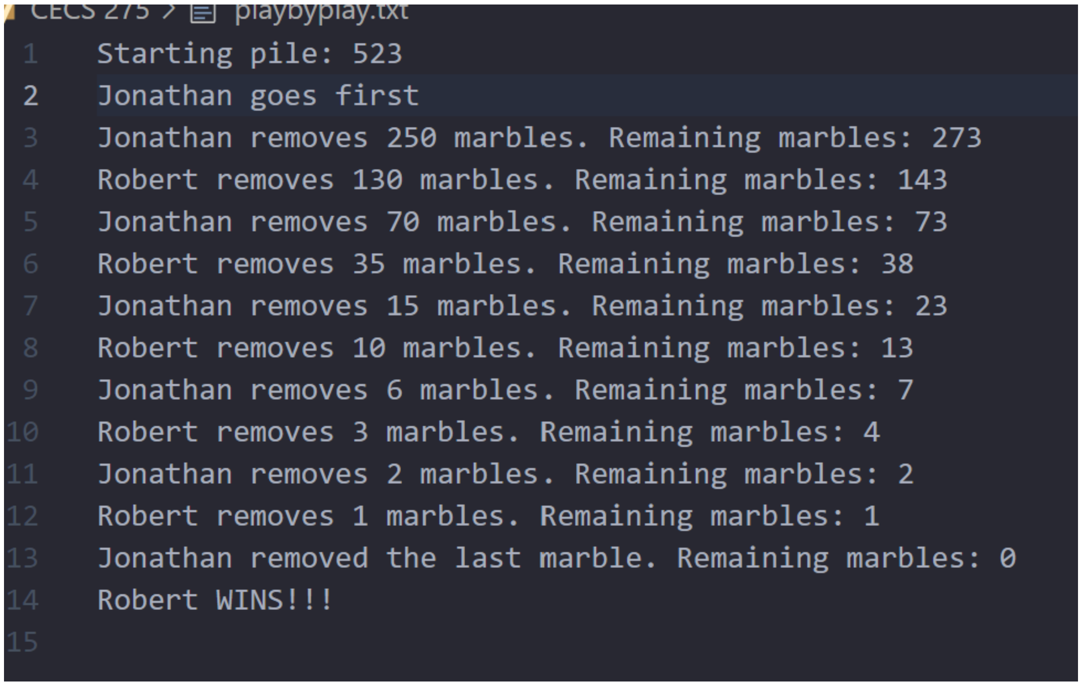


```
Starting pile: 63
jon goes first
jon removes 30 marbles. Remaining marbles: 33
Bot removes 7 marbles. Remaining marbles: 26
jon removes 13 marbles. Remaining marbles: 13
Bot removes 5 marbles. Remaining marbles: 8
jon removes 4 marbles. Remaining marbles: 4
Bot removes 2 marbles. Remaining marbles: 2
jon removes 1 marbles. Remaining marbles: 1
Bot removed the last marble. Remaining marbles: 0
jon WINS!!!
```

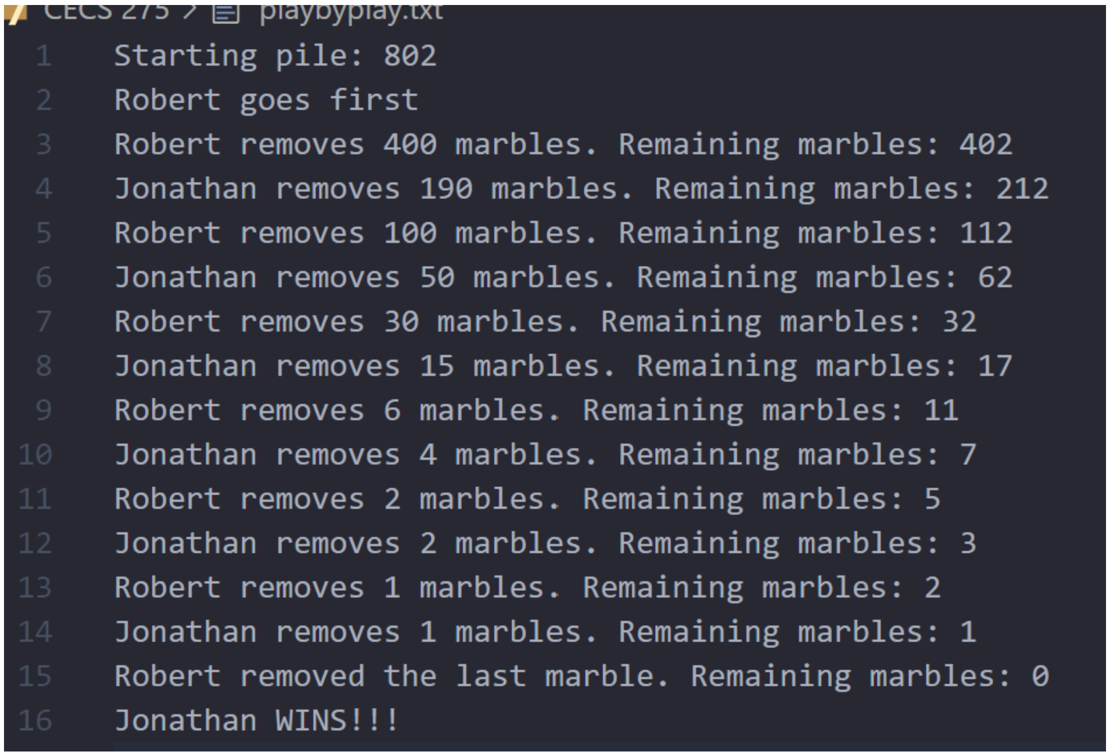


```
Starting pile: 483
jon goes first
jon removes 200 marbles. Remaining marbles: 283
Bot removes 28 marbles. Remaining marbles: 255
jon removes 125 marbles. Remaining marbles: 130
Bot removes 3 marbles. Remaining marbles: 127
jon removes 60 marbles. Remaining marbles: 67
Bot removes 4 marbles. Remaining marbles: 63
jon removes 30 marbles. Remaining marbles: 33
Bot removes 2 marbles. Remaining marbles: 31
jon removes 15 marbles. Remaining marbles: 16
Bot removes 1 marbles. Remaining marbles: 15
jon removes 7 marbles. Remaining marbles: 8
Bot removes 1 marbles. Remaining marbles: 7
jon removes 3 marbles. Remaining marbles: 4
Bot removes 1 marbles. Remaining marbles: 3
jon removes 1 marbles. Remaining marbles: 2
Bot removes 1 marbles. Remaining marbles: 1
jon removed the last marble. Remaining marbles: 0
Bot WINS!!!
```

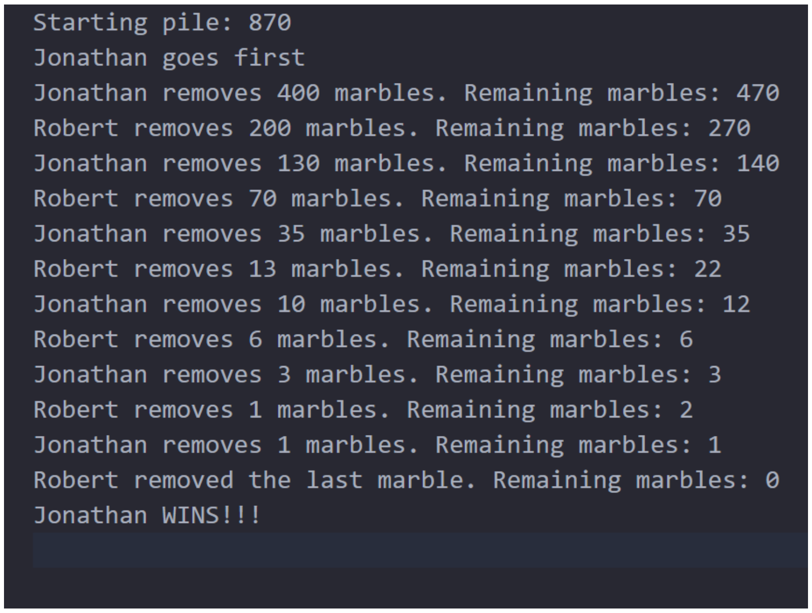
Against Human:

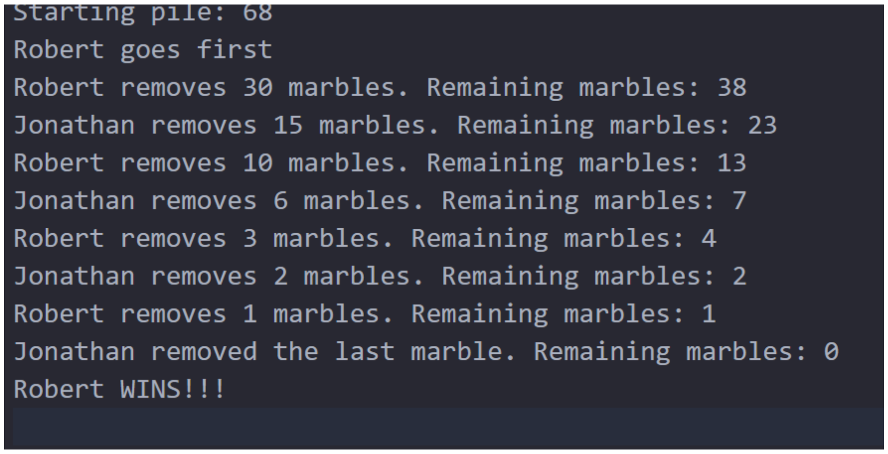


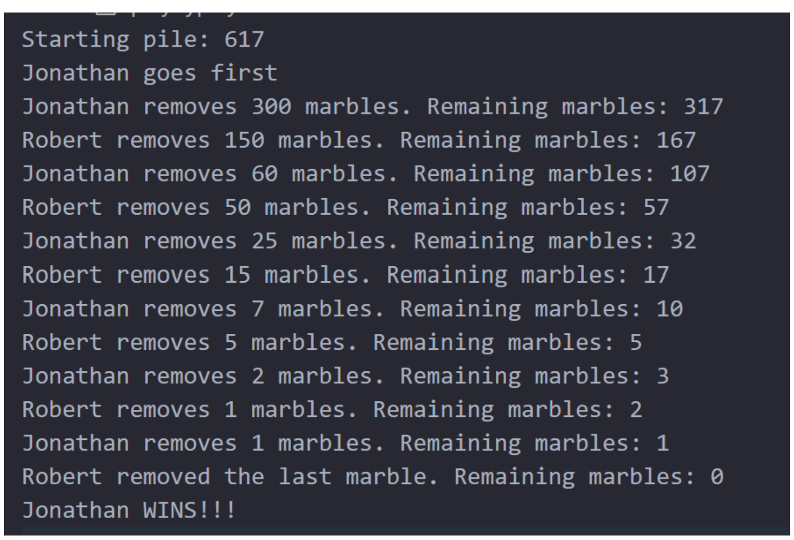
```
CECS 275 > playbyplay.txt
Starting pile: 523
Jonathan goes first
Jonathan removes 250 marbles. Remaining marbles: 273
Robert removes 130 marbles. Remaining marbles: 143
Jonathan removes 70 marbles. Remaining marbles: 73
Robert removes 35 marbles. Remaining marbles: 38
Jonathan removes 15 marbles. Remaining marbles: 23
Robert removes 10 marbles. Remaining marbles: 13
Jonathan removes 6 marbles. Remaining marbles: 7
Robert removes 3 marbles. Remaining marbles: 4
Jonathan removes 2 marbles. Remaining marbles: 2
Robert removes 1 marbles. Remaining marbles: 1
Jonathan removed the last marble. Remaining marbles: 0
Robert WINS!!!
```

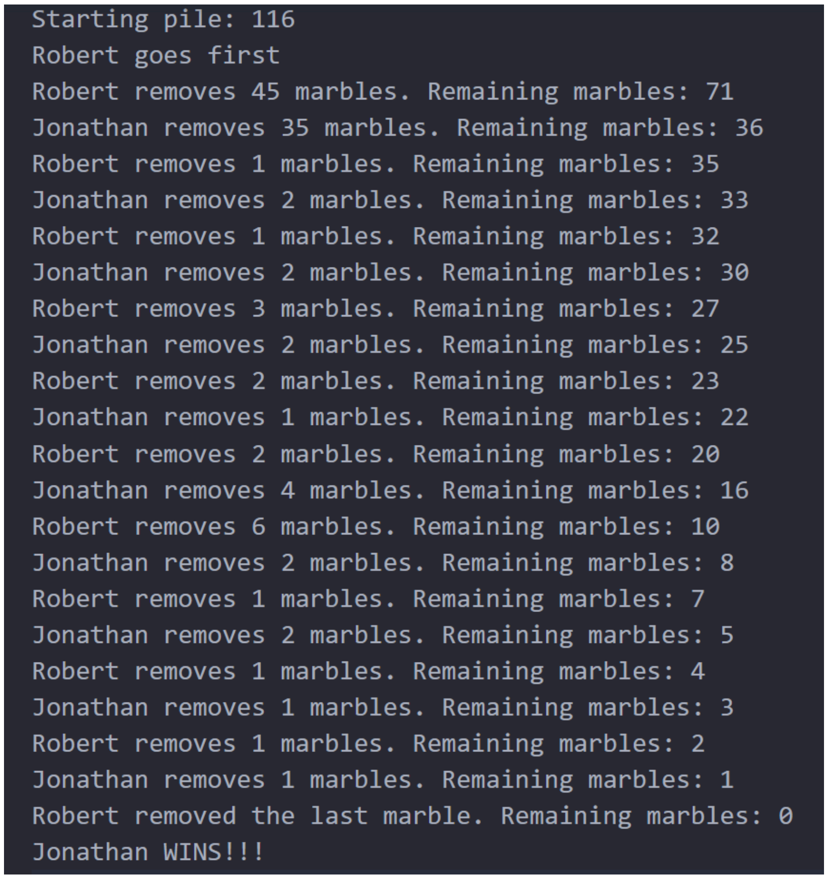


```
CECS 275 > playbyplay.txt
Starting pile: 802
Robert goes first
Robert removes 400 marbles. Remaining marbles: 402
Jonathan removes 190 marbles. Remaining marbles: 212
Robert removes 100 marbles. Remaining marbles: 112
Jonathan removes 50 marbles. Remaining marbles: 62
Robert removes 30 marbles. Remaining marbles: 32
Jonathan removes 15 marbles. Remaining marbles: 17
Robert removes 6 marbles. Remaining marbles: 11
Jonathan removes 4 marbles. Remaining marbles: 7
Robert removes 2 marbles. Remaining marbles: 5
Jonathan removes 2 marbles. Remaining marbles: 3
Robert removes 1 marbles. Remaining marbles: 2
Jonathan removes 1 marbles. Remaining marbles: 1
Robert removed the last marble. Remaining marbles: 0
Jonathan WINS!!!
```


```
Starting pile: 870
Jonathan goes first
Jonathan removes 400 marbles. Remaining marbles: 470
Robert removes 200 marbles. Remaining marbles: 270
Jonathan removes 130 marbles. Remaining marbles: 140
Robert removes 70 marbles. Remaining marbles: 70
Jonathan removes 35 marbles. Remaining marbles: 35
Robert removes 13 marbles. Remaining marbles: 22
Jonathan removes 10 marbles. Remaining marbles: 12
Robert removes 6 marbles. Remaining marbles: 6
Jonathan removes 3 marbles. Remaining marbles: 3
Robert removes 1 marbles. Remaining marbles: 2
Jonathan removes 1 marbles. Remaining marbles: 1
Robert removed the last marble. Remaining marbles: 0
Jonathan WINS!!!
```


```
Starting pile: 68
Robert goes first
Robert removes 30 marbles. Remaining marbles: 38
Jonathan removes 15 marbles. Remaining marbles: 23
Robert removes 10 marbles. Remaining marbles: 13
Jonathan removes 6 marbles. Remaining marbles: 7
Robert removes 3 marbles. Remaining marbles: 4
Jonathan removes 2 marbles. Remaining marbles: 2
Robert removes 1 marbles. Remaining marbles: 1
Jonathan removed the last marble. Remaining marbles: 0
Robert WINS!!!
```

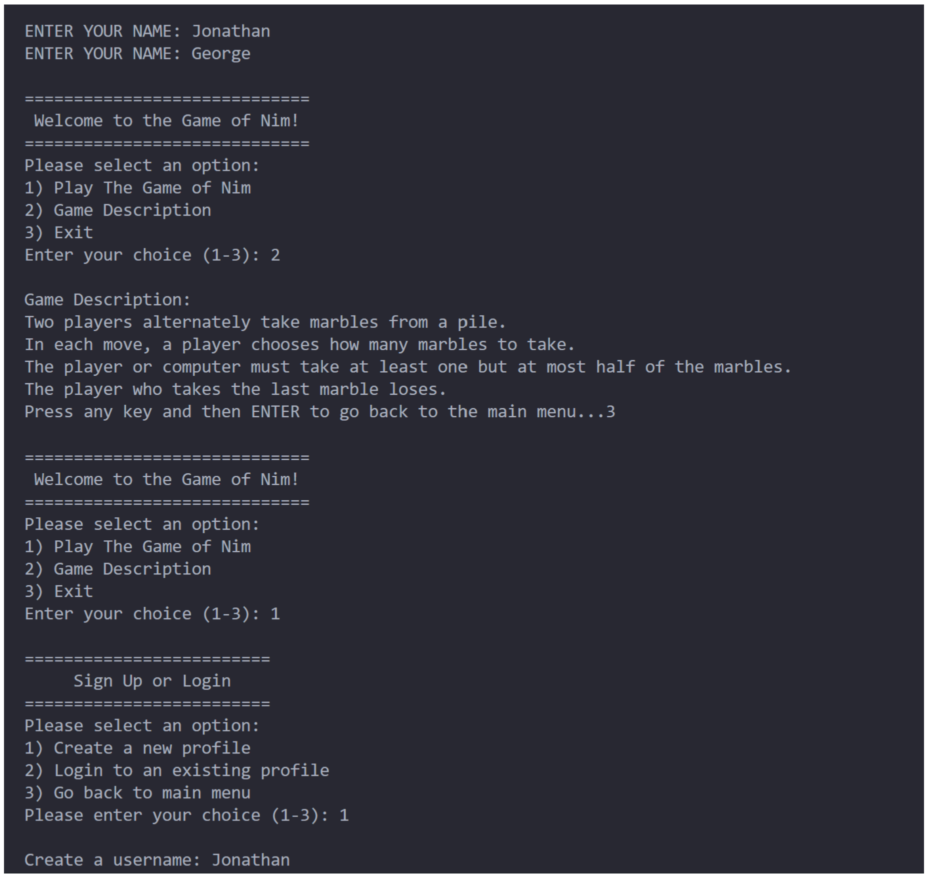

```
Starting pile: 617
Jonathan goes first
Jonathan removes 300 marbles. Remaining marbles: 317
Robert removes 150 marbles. Remaining marbles: 167
Jonathan removes 60 marbles. Remaining marbles: 107
Robert removes 50 marbles. Remaining marbles: 57
Jonathan removes 25 marbles. Remaining marbles: 32
Robert removes 15 marbles. Remaining marbles: 17
Jonathan removes 7 marbles. Remaining marbles: 10
Robert removes 5 marbles. Remaining marbles: 5
Jonathan removes 2 marbles. Remaining marbles: 3
Robert removes 1 marbles. Remaining marbles: 2
Jonathan removes 1 marbles. Remaining marbles: 1
Robert removed the last marble. Remaining marbles: 0
Jonathan WINS!!!
```



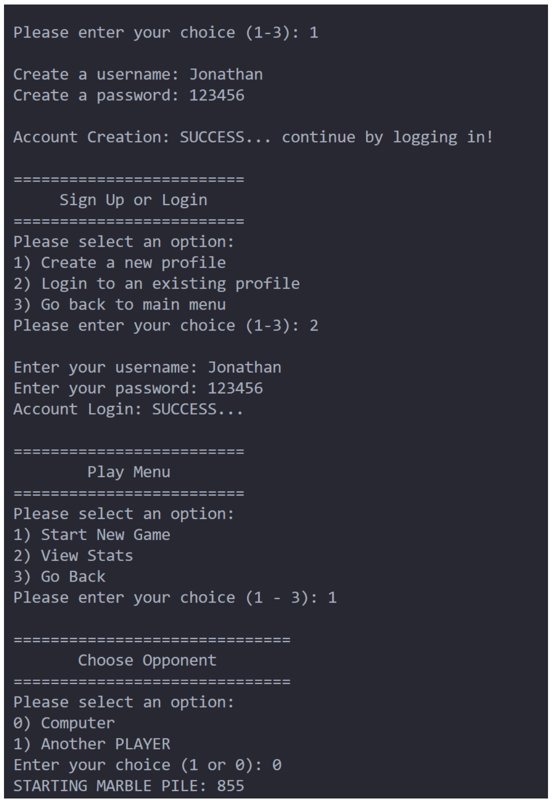
```
Starting pile: 116
Robert goes first
Robert removes 45 marbles. Remaining marbles: 71
Jonathan removes 35 marbles. Remaining marbles: 36
Robert removes 1 marbles. Remaining marbles: 35
Jonathan removes 2 marbles. Remaining marbles: 33
Robert removes 1 marbles. Remaining marbles: 32
Jonathan removes 2 marbles. Remaining marbles: 30
Robert removes 3 marbles. Remaining marbles: 27
Jonathan removes 2 marbles. Remaining marbles: 25
Robert removes 2 marbles. Remaining marbles: 23
Jonathan removes 1 marbles. Remaining marbles: 22
Robert removes 2 marbles. Remaining marbles: 20
Jonathan removes 4 marbles. Remaining marbles: 16
Robert removes 6 marbles. Remaining marbles: 10
Jonathan removes 2 marbles. Remaining marbles: 8
Robert removes 1 marbles. Remaining marbles: 7
Jonathan removes 2 marbles. Remaining marbles: 5
Robert removes 1 marbles. Remaining marbles: 4
Jonathan removes 1 marbles. Remaining marbles: 3
Robert removes 1 marbles. Remaining marbles: 2
Jonathan removes 1 marbles. Remaining marbles: 1
Robert removed the last marble. Remaining marbles: 0
Jonathan WINS!!!
```

Terminal/Output:

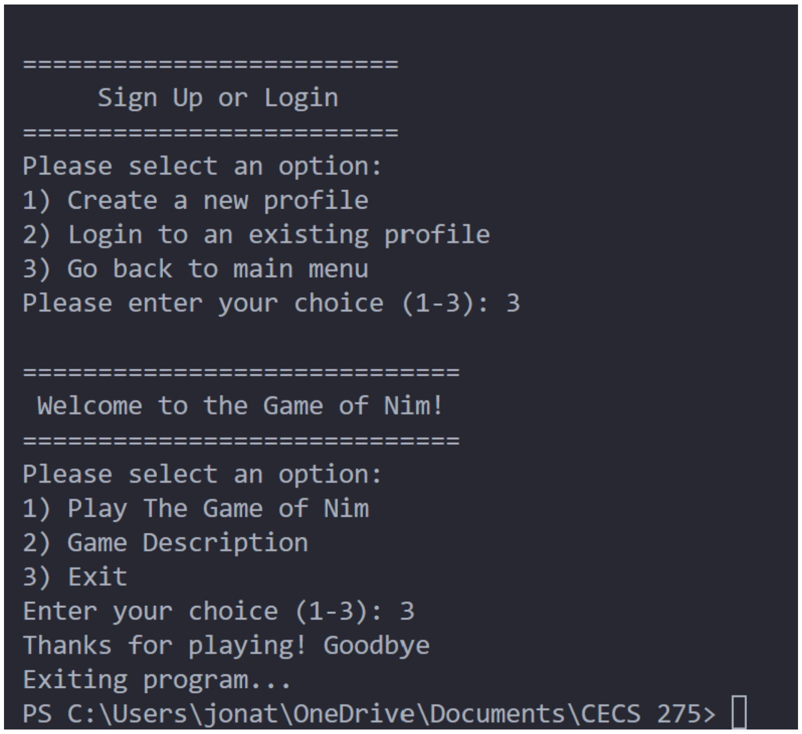
Menus:



```
ENTER YOUR NAME: Jonathan
ENTER YOUR NAME: George

=============================
 Welcome to the Game of Nim!
=============================
Please select an option:
1) Play The Game of Nim
2) Game Description
3) Exit
Enter your choice (1-3): 2

Game Description:
Two players alternately take marbles from a pile.
In each move, a player chooses how many marbles to take.
The player or computer must take at least one but at most half of the marbles.
The player who takes the last marble loses.
Press any key and then ENTER to go back to the main menu...3

=============================
 Welcome to the Game of Nim!
=============================
Please select an option:
1) Play The Game of Nim
2) Game Description
3) Exit
Enter your choice (1-3): 1

=========================
     Sign Up or Login
=========================
Please select an option:
1) Create a new profile
2) Login to an existing profile
3) Go back to main menu
Please enter your choice (1-3): 1

Create a username: Jonathan
```



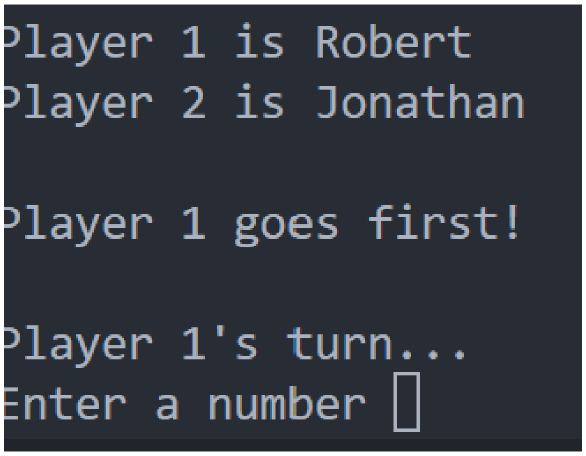
```
Please enter your choice (1-3): 1

Create a username: Jonathan
Create a password: 123456

Account Creation: SUCCESS... continue by logging in!

=========================
     Sign Up or Login
=========================
Please select an option:
1) Create a new profile
2) Login to an existing profile
3) Go back to main menu
Please enter your choice (1-3): 2

Enter your username: Jonathan
Enter your password: 123456
Account Login: SUCCESS...

=========================
        Play Menu
=========================
Please select an option:
1) Start New Game
2) View Stats
3) Go Back
Please enter your choice (1 - 3): 1

==============================
       Choose Opponent
==============================
Please select an option:
0) Computer
1) Another PLAYER
Enter your choice (1 or 0): 0
STARTING MARBLE PILE: 855
```



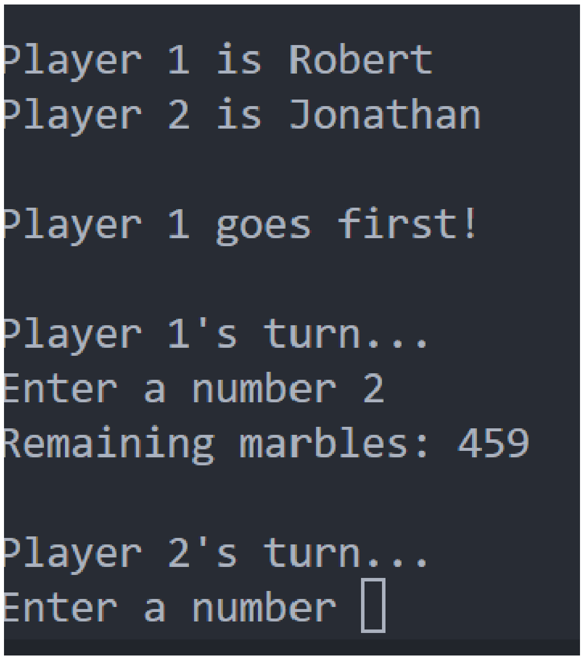
```
=========================
     Sign Up or Login
=========================
Please select an option:
1) Create a new profile
2) Login to an existing profile
3) Go back to main menu
Please enter your choice (1-3): 3

=============================
 Welcome to the Game of Nim!
=============================
Please select an option:
1) Play The Game of Nim
2) Game Description
3) Exit
Enter your choice (1-3): 3
Thanks for playing! Goodbye
Exiting program...
PS C:\Users\jonat\OneDrive\Documents\CECS 275>
```

Ex.) Against Human

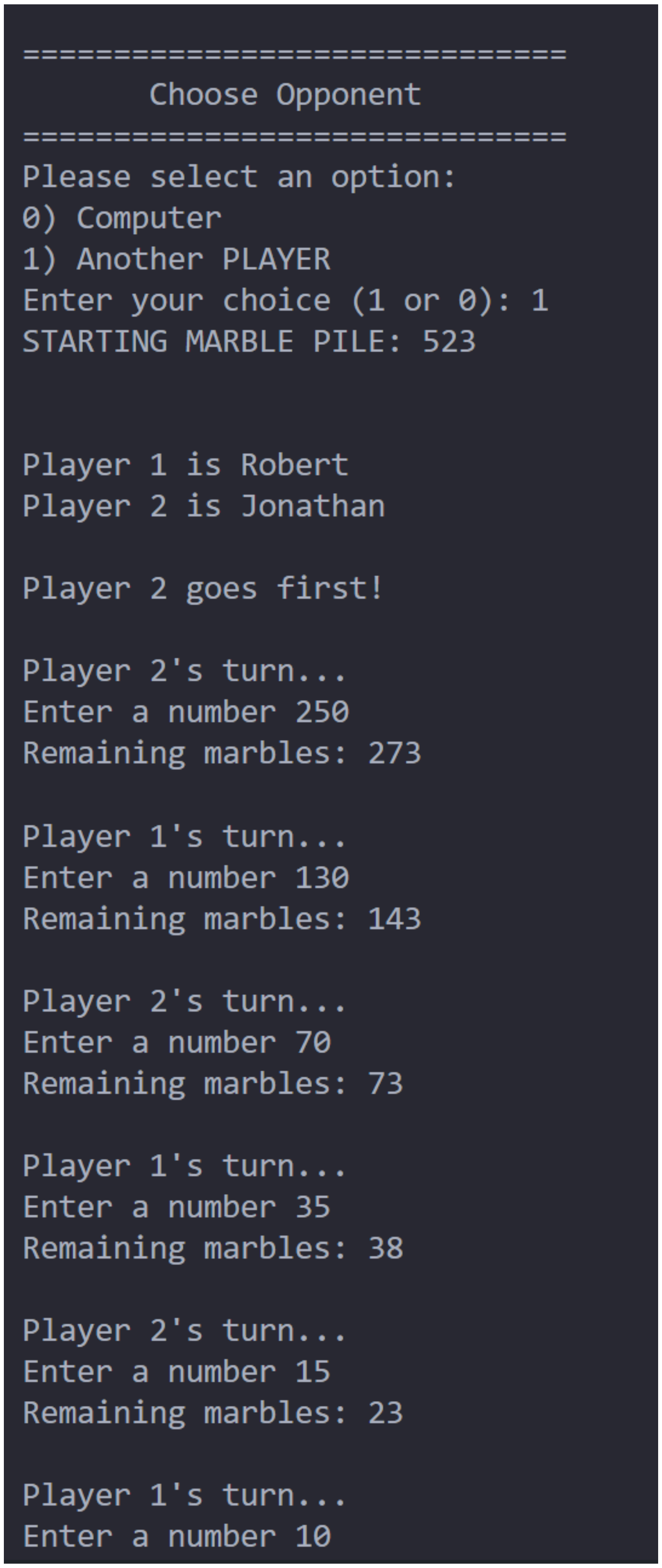
Asks both players for inputs:



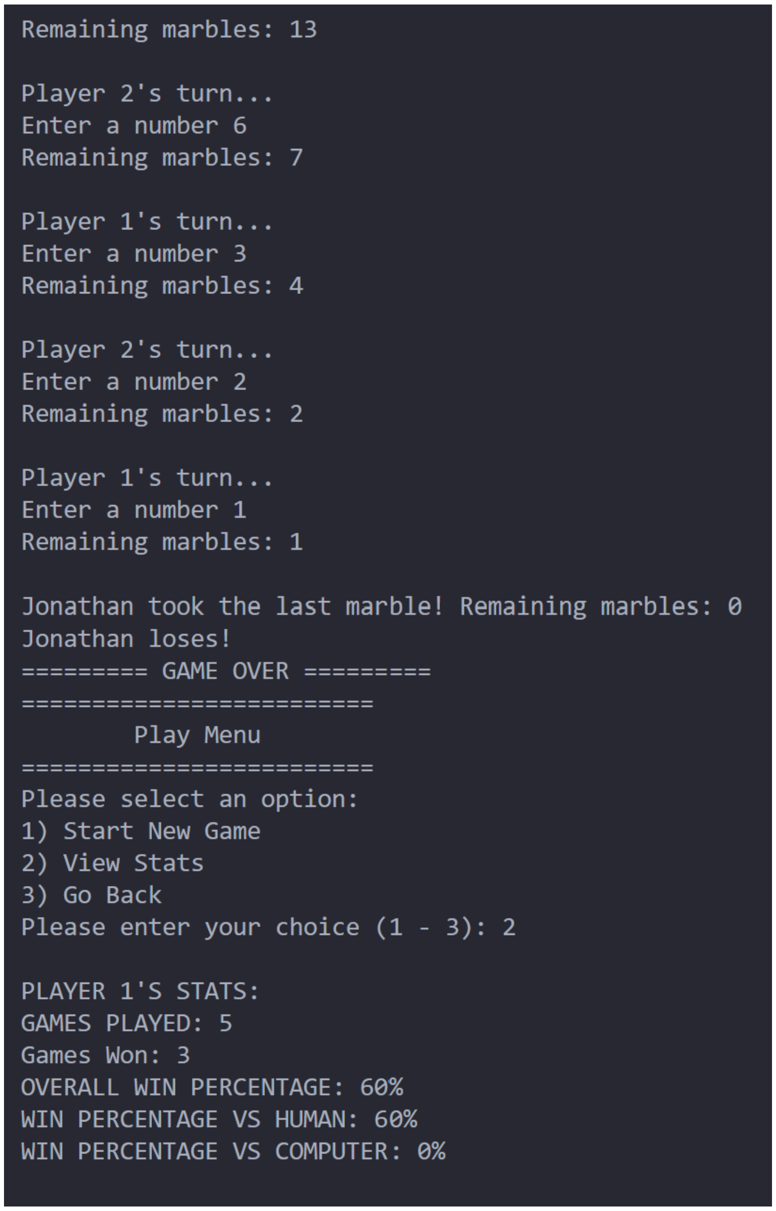
```
Player 1 is Robert
Player 2 is Jonathan

Player 1 goes first!

Player 1's turn...
Enter a number
```



```
Player 1 is Robert
Player 2 is Jonathan

Player 1 goes first!

Player 1's turn...
Enter a number 2
Remaining marbles: 459

Player 2's turn...
Enter a number
```

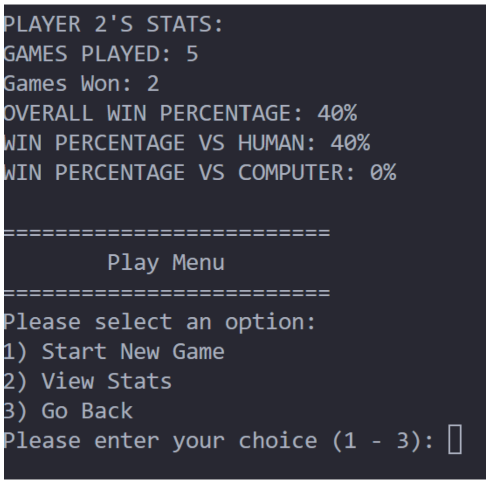


```
==============================
       Choose Opponent
==============================
Please select an option:
0) Computer
1) Another PLAYER
Enter your choice (1 or 0): 1
STARTING MARBLE PILE: 523


Player 1 is Robert
Player 2 is Jonathan

Player 2 goes first!

Player 2's turn...
Enter a number 250
Remaining marbles: 273

Player 1's turn...
Enter a number 130
Remaining marbles: 143

Player 2's turn...
Enter a number 70
Remaining marbles: 73

Player 1's turn...
Enter a number 35
Remaining marbles: 38

Player 2's turn...
Enter a number 15
Remaining marbles: 23

Player 1's turn...
Enter a number 10
```

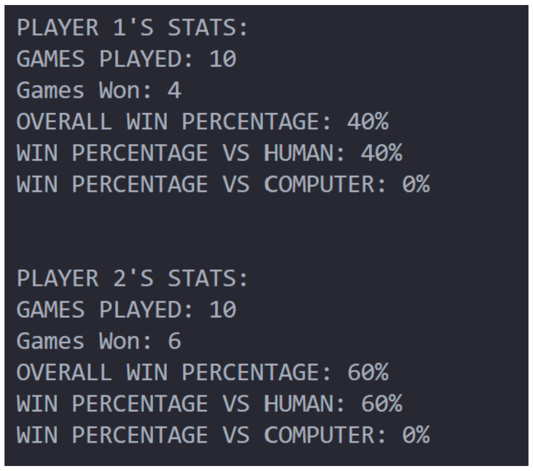


```
Remaining marbles: 13

Player 2's turn...
Enter a number 6
Remaining marbles: 7

Player 1's turn...
Enter a number 3
Remaining marbles: 4

Player 2's turn...
Enter a number 2
Remaining marbles: 2

Player 1's turn...
Enter a number 1
Remaining marbles: 1

Jonathan took the last marble! Remaining marbles: 0
Jonathan loses!
========= GAME OVER =========
=========================
        Play Menu
=========================
Please select an option:
1) Start New Game
2) View Stats
3) Go Back
Please enter your choice (1 - 3): 2

PLAYER 1'S STATS:
GAMES PLAYED: 5
Games Won: 3
OVERALL WIN PERCENTAGE: 60%
WIN PERCENTAGE VS HUMAN: 60%
WIN PERCENTAGE VS COMPUTER: 0%
```

Stats for PVP:

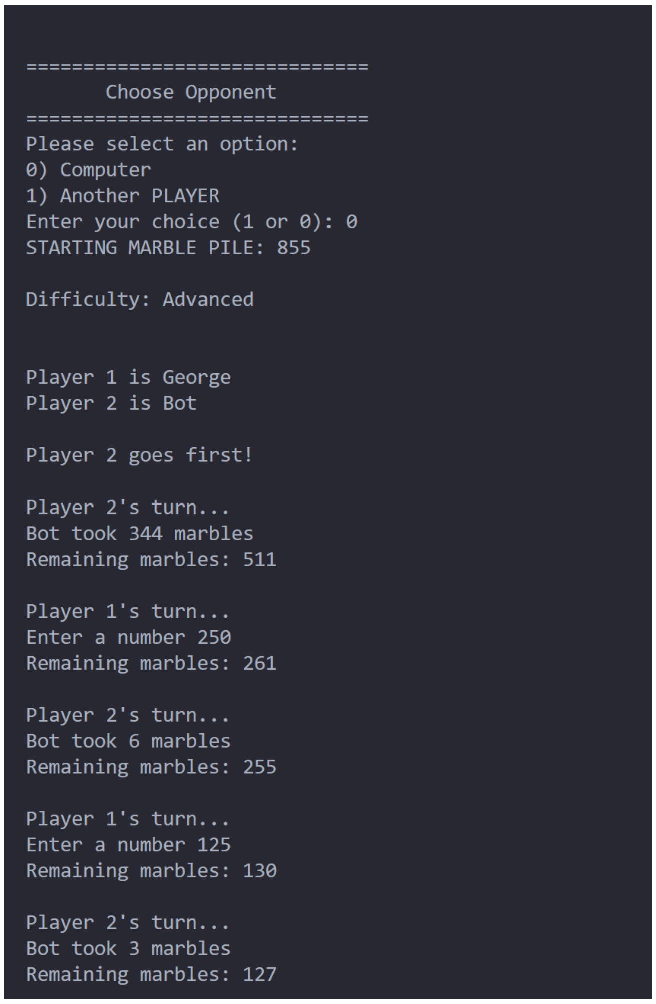


```
PLAYER 2'S STATS:
GAMES PLAYED: 5
Games Won: 2
OVERALL WIN PERCENTAGE: 40%
WIN PERCENTAGE VS HUMAN: 40%
WIN PERCENTAGE VS COMPUTER: 0%

=========================
        Play Menu
=========================
Please select an option:
1) Start New Game
2) View Stats
3) Go Back
Please enter your choice (1 - 3):
```

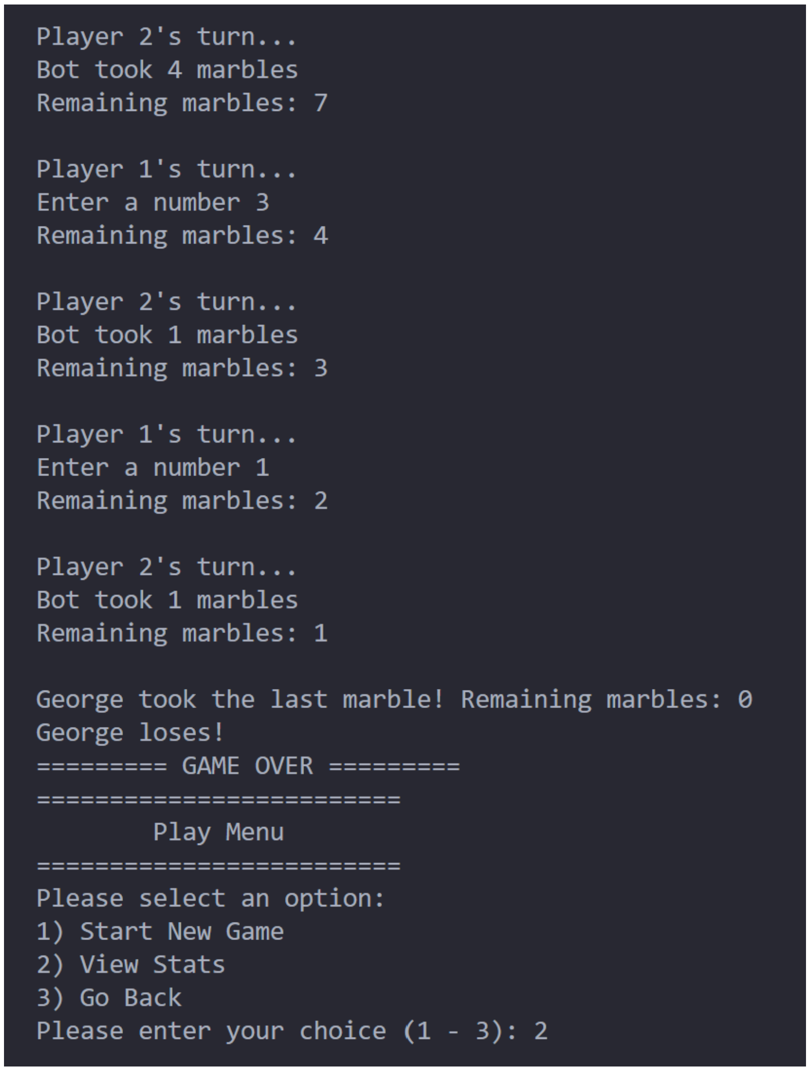


```
PLAYER 1'S STATS:
GAMES PLAYED: 10
Games Won: 4
OVERALL WIN PERCENTAGE: 40%
WIN PERCENTAGE VS HUMAN: 40%
WIN PERCENTAGE VS COMPUTER: 0%


PLAYER 2'S STATS:
GAMES PLAYED: 10
Games Won: 6
OVERALL WIN PERCENTAGE: 60%
WIN PERCENTAGE VS HUMAN: 60%
WIN PERCENTAGE VS COMPUTER: 0%
```

Ex.) Against Computer:

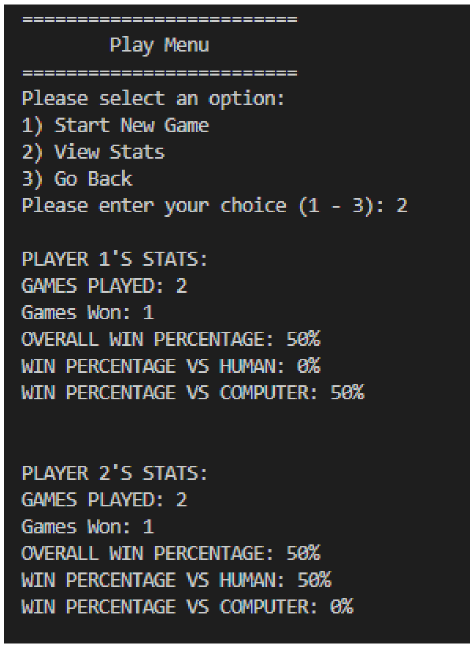


```
==============================
       Choose Opponent
==============================
Please select an option:
0) Computer
1) Another PLAYER
Enter your choice (1 or 0): 0
STARTING MARBLE PILE: 855

Difficulty: Advanced


Player 1 is George
Player 2 is Bot

Player 2 goes first!

Player 2's turn...
Bot took 344 marbles
Remaining marbles: 511

Player 1's turn...
Enter a number 250
Remaining marbles: 261

Player 2's turn...
Bot took 6 marbles
Remaining marbles: 255

Player 1's turn...
Enter a number 125
Remaining marbles: 130

Player 2's turn...
Bot took 3 marbles
Remaining marbles: 127
```



```
Player 2's turn...
Bot took 4 marbles
Remaining marbles: 7

Player 1's turn...
Enter a number 3
Remaining marbles: 4

Player 2's turn...
Bot took 1 marbles
Remaining marbles: 3

Player 1's turn...
Enter a number 1
Remaining marbles: 2

Player 2's turn...
Bot took 1 marbles
Remaining marbles: 1

George took the last marble! Remaining marbles: 0
George loses!
========= GAME OVER =========
=========================
        Play Menu
=========================
Please select an option:
1) Start New Game
2) View Stats
3) Go Back
Please enter your choice (1 - 3): 2
```

Stats against Computer:



```
=========================
        Play Menu
=========================
Please select an option:
1) Start New Game
2) View Stats
3) Go Back
Please enter your choice (1 - 3): 2

PLAYER 1'S STATS:
GAMES PLAYED: 2
Games Won: 1
OVERALL WIN PERCENTAGE: 50%
WIN PERCENTAGE VS HUMAN: 0%
WIN PERCENTAGE VS COMPUTER: 50%


PLAYER 2'S STATS:
GAMES PLAYED: 2
Games Won: 1
OVERALL WIN PERCENTAGE: 50%
WIN PERCENTAGE VS HUMAN: 50%
WIN PERCENTAGE VS COMPUTER: 0%
```