Course:

CIS-17B

Title:

Online Connect 4

Assignment:

Project Reports

Due Date

May 12, 2024

Authors:

Aleksandar Videv, Amare Terrell, Albert Ruiz, Anthony Nguyen, Cristian Magana, Francisco Sanchez, Janaye Jackson,

Kelby Knight, Kyle Riebeling, Patrick Pascual,

Rafaan Hyder, Ryan Westfall.

Table of Contents

1. Introduction  
2. Project Overview  
 2.1 Table of Project Development  
3. Description  
4. Gantt Chart  
5. UML  
6. Pseudo Code  
7. Group Information  
 7.1 Time and location of meetings  
 7.2 Lists of members, responsibilities, and assignments  
8. Code  
9. Proof of working code

**1. Introduction**

The game of Connect 4 has been a popular pastime for many years, often played with physical tokens and a vertical board. I have developed a digital version of Connect 4 in C++ in this project. This version allows players to compete against each other and includes an option to play against a computer-controlled opponent utilizing artificial intelligence. The game saves user data and game statistics in binary files, allowing players to keep track of their progress over time. This implementation aims to enhance user engagement through strategic gameplay and persistent data management.

**2. Project Overview**

The software is structured around several key C++ classes that handle different aspects of the game, such as managing the game board, processing player interactions, and storing user data. These components are:

**BinaryInterface**: This interface facilitates all operations related to user data, including login procedures and data persistence, using a binary file system.

**Board**: Manages the game board's state, keeping track of each cell within a 6x7 grid.

**Game**: Orchestrates the gameplay, including initializing the game, responding to player moves, executing computer moves using AI algorithms, and determining the game's outcome.

**User**: Maintains detailed user information and game statistics, with functionality to manage and store this data effectively.

This modular architecture helps keep the code organized and ensures that each part is easily manageable and can be updated independently.

2.1 Table of Project Development

| Version | Lines | Board development | Game Logic | User interface | Binary Interface | Other Files |
| --- | --- | --- | --- | --- | --- | --- |
| V1 | 141 | ✖ | ✖ | ✖ | ✖ | 4 |
| V2.0 | 333 | ✔ | ✔ | ✖ | ✖ | 2 |
| V2.1 | 338 | ✔ | ✔ | ✖ | ✖ | 4 |
| V2.2 | 405 | ✔ | ✔ | ✖ | ✖ | 3 |
| V2.3 | 453 | ✔ | ✔ | ✖ | ✖ | 4 |
| V3.0 | 759 | ✔ | ✔ | ✔ | ✔ | 4 |
| V3.1 | 843 | ✔ | ✔ | ✔ | ✔ | 4 |
| V3.2 | 823 | ✔ | ✔ | ✔ | ✔ | 4 |
| V3.3 | 808 | ✔ | ✔ | ✔ | ✔ | 4 |
| V3.4 | 1,012 | ✔ | ✔ | ✔ | ✔ | 4 |

**3. Description**

**BinaryInterface Class**

This class is crucial for managing user data interactions, particularly for logging in users and handling their game records. It works closely with a binary file to store and retrieve user data efficiently, ensuring that player statistics are maintained accurately and securely.

**Board Class**

The BOARD class represents the Connect 4 board, the foundation of the game interface. It consists of a two-dimensional grid where players’ moves are registered and stored, allowing the system to check for winning combinations and valid moves.

**Game**

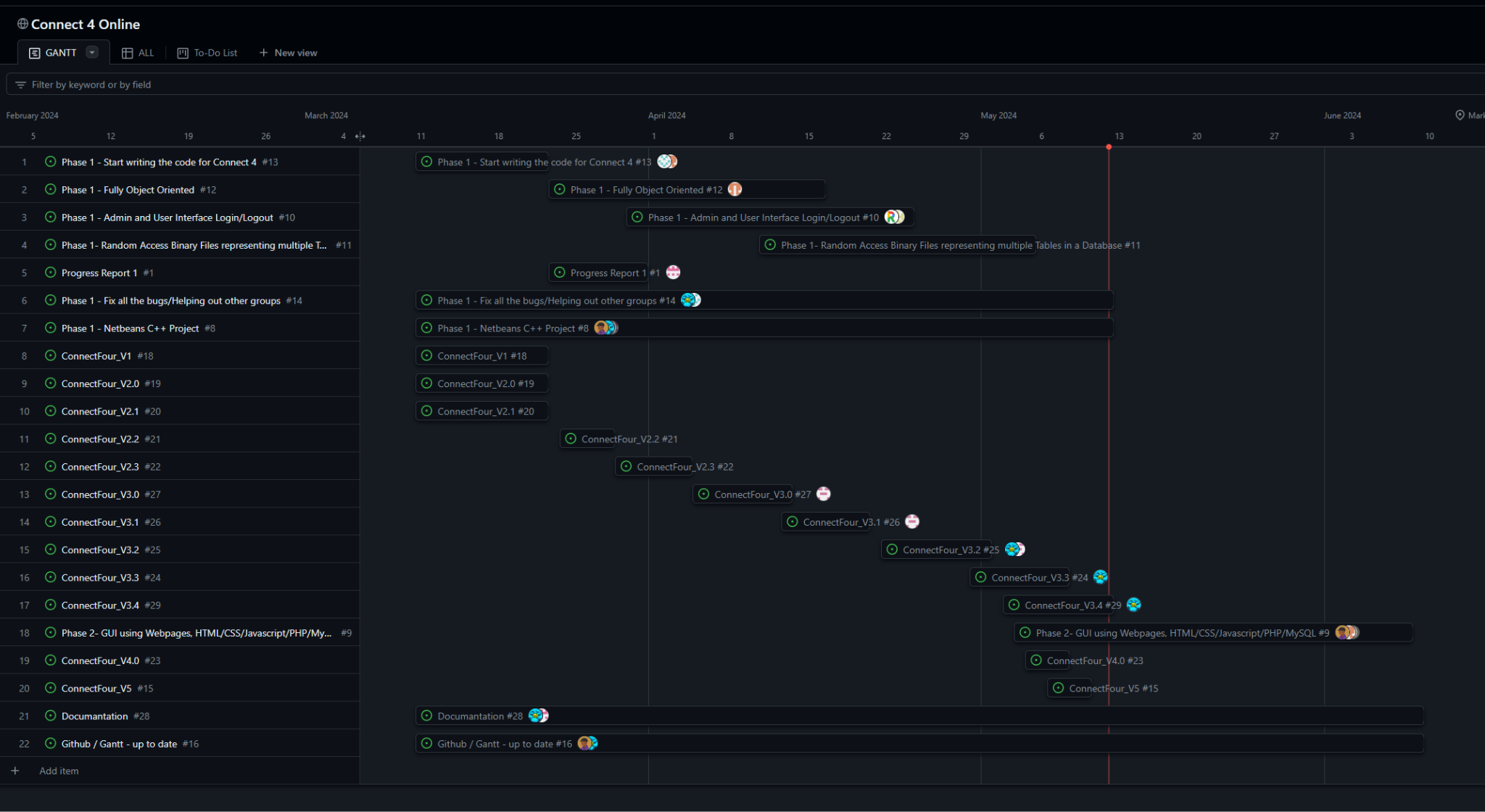
The Game class is central to the implementation, handling the core logic required to play Connect 4. It manages game sessions, processes player inputs, and executes AI strategies. The AI logic uses a minimax algorithm enhanced with alpha-beta pruning to make intelligent decisions. The class also updates players on the game state through a simple console-based interface, making the game interactive and engaging.

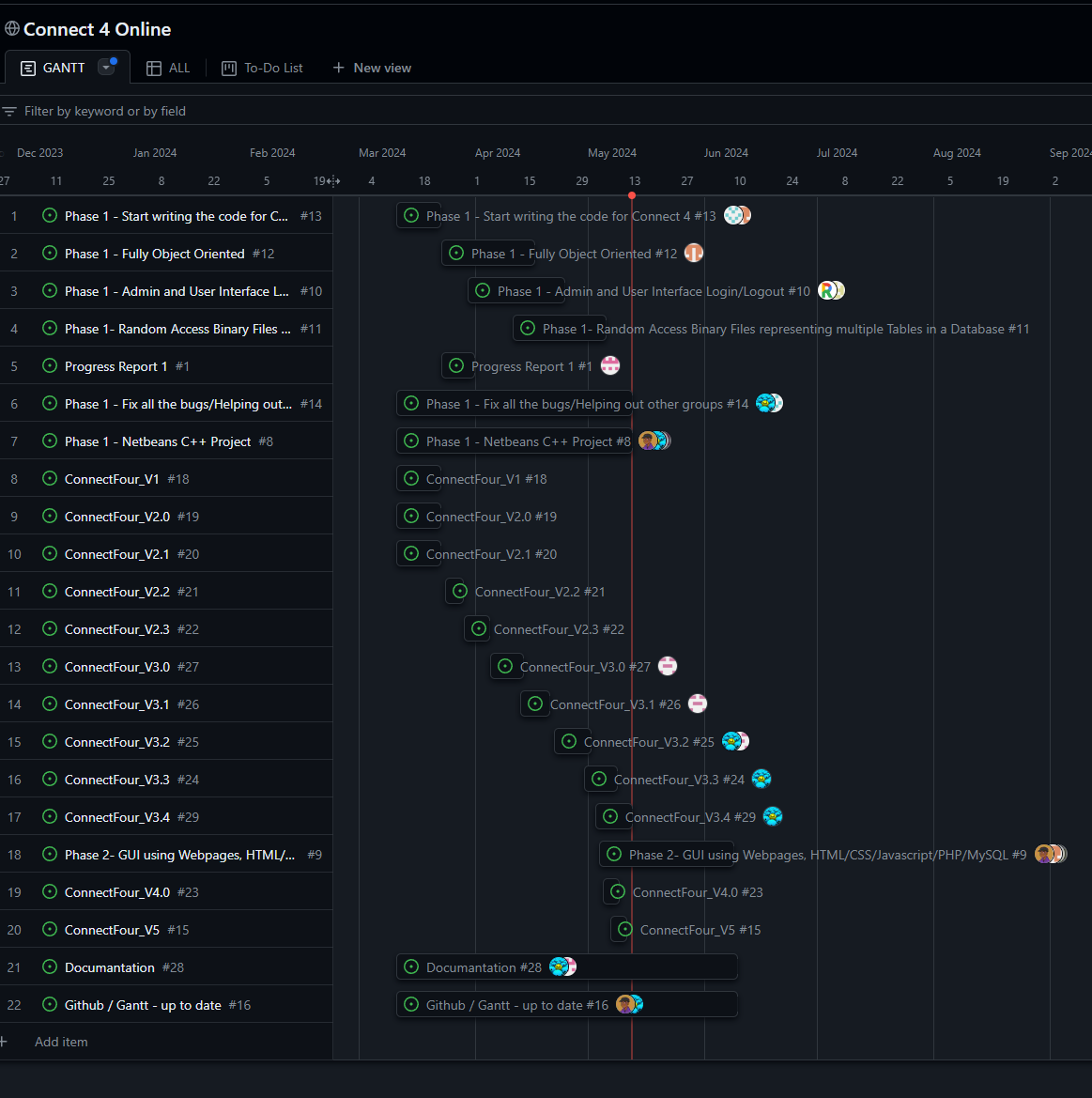
**User Class**

The User class encapsulates all player-related information and stores individual player profiles, including usernames and records of past games. It provides essential methods for manipulating this data, such as adding new game results and interfacing with the binary file system for data persistence.

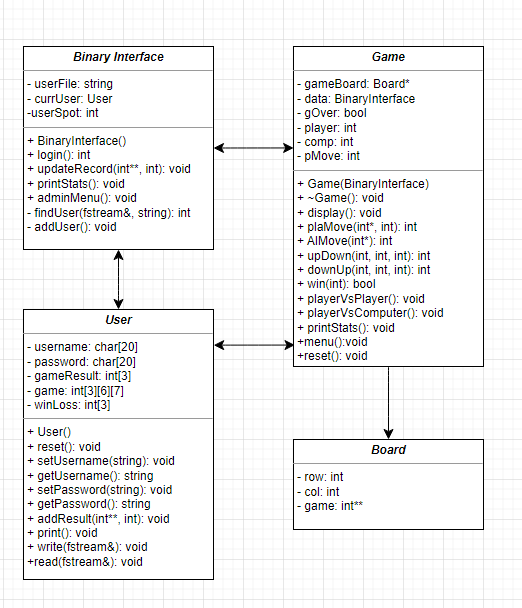
**4. Gantt Chart**

The Gantt Chart can be found on [GitHub](https://github.com/users/4mxr3/projects/2/views/5).





**5. UML**



**6. Pseudo-Code:**

1. Main
2. Login
   1. If Admin Login (Result = 1)
   2. Else if User Login
      1. New User? Create new account and save (Result = 0)
      2. Returning User? Ask for password (Match = 0, Else = -1)
   3. Else (Result = -1)
3. Game Initialization
   1. Game class
   2. User Interactions
4. Binary and file operations
5. User management
   1. User class
      1. Stores user details
6. Game Play Loop
   1. Display Board
   2. Prompt Player to make a move
      1. Update board based on move
   3. Check for a win or draw
   4. If the game is over display the winner or declare a draw
      1. Otherwise, Switch to the next player
7. Ending game
   1. Save game state
   2. Update scores
   3. Return to log in or exit

| Line count: | Login Handling | Game logic | User Management | Binary Operations |
| --- | --- | --- | --- | --- |
| 823 | ✔ | ✔ | ✔ | ✔ |

**7. Group Information**

Our group utilized a Gantt chart to allocate and track tasks as part of our project management strategy. Below is a detailed breakdown of each member's contributions and the logistical aspects of our meetings.

**7.1 Time and Location of Meetings**

To coordinate our schedules efficiently, we set up a Google Calendar. This helped us identify the best times for all members to meet. The calendar is accessible [here](https://calendar.google.com/calendar/u/0/r?cid=NmY3MGFhOWM3NjJhYjgwMTdhMzg1Yjg1MjkyYzU5MDIwOWEyZjMxZjhhYjVhMDllNDljMTg4ODkwNTUwNDkxNkBncm91cC5jYWxlbmRhci5nb29nbGUuY29t). Based on our availability, we decided to schedule meetings twice a week, with flexibility depending on our workload.

**Scheduled Meeting Times:**Tuesdays and Sundays from 5 PM to 12 AM

**7.2 List of Members, Responsibilities, and Assignments**

**Aleksandar Videv:** Managed the project using a Gantt chart and maintained GitHub. Developed version 3.3, addressing input and menu errors, and version 3.4, improving AI with minimax. Contributed to the documentation.

**Amare Terrell:** Update the Github Project (GANTT Chart, To-Do List), Assistance with Documentation, and later work on the JavaScript & PHP portions.

**Anthony Nguyen:** Documentation and creation of UML/FlowChart

**Cristian Magana:**

**Francisco Sanchez:** Added the menu in v2.2 and implemented the player vs player mode and moved some logic to the Game class to keep main.cpp cleaner. Fixed the output bug where the board was printing twice for AI and player.

**Janaye Jackson:** Reconfigured the already written code in V1 into object-oriented programming, created the AI, and added more conditions to make it more complicated. Debugged some logic I wrote that was inconsistent. Also converted the project into basic javascript, have not implemented database or user/admin.

**Kelby Knight:** Organizing group meetings & Documentation

**Kyle Riebeling:** User accounts and storing their data in binary files. Admin functions with default admin account (username=”admin”, pass=”password”)

**Patrick Pascual:** Set up initial base codeline, or first verson as well as helped with the documentation of the code.

**Rafaan Hyder:**

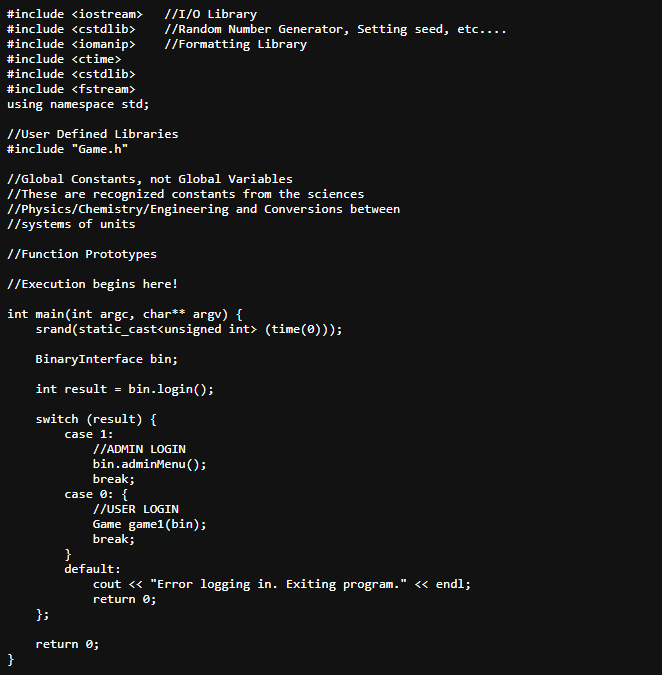
**Ryan Westfall:** Converted project into php

**8. Code**

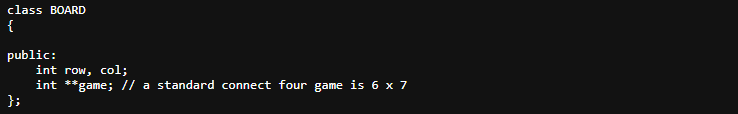
All versions of the code are on [Github](https://github.com/4mxr3/Connect4ON), Version 3.4 is the final C++ version.

**Images of Code :**

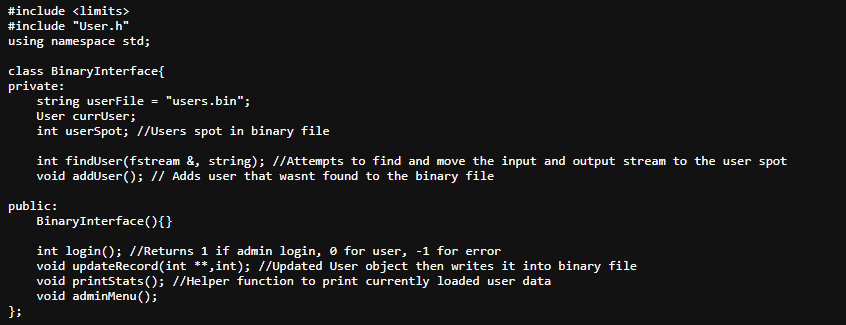
**Main:**

****

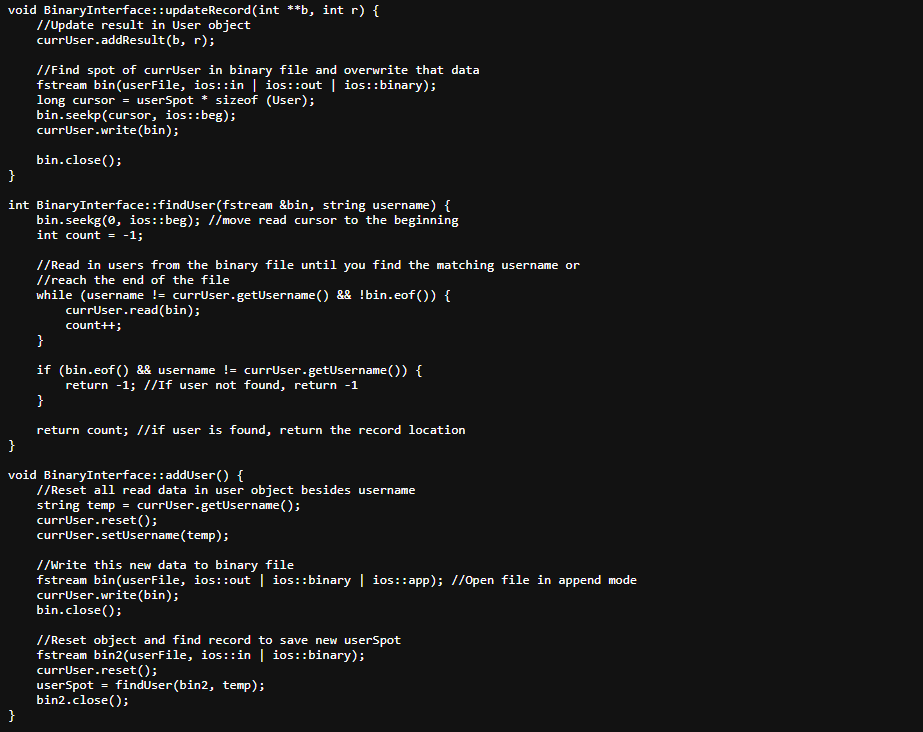
**Board:**

****

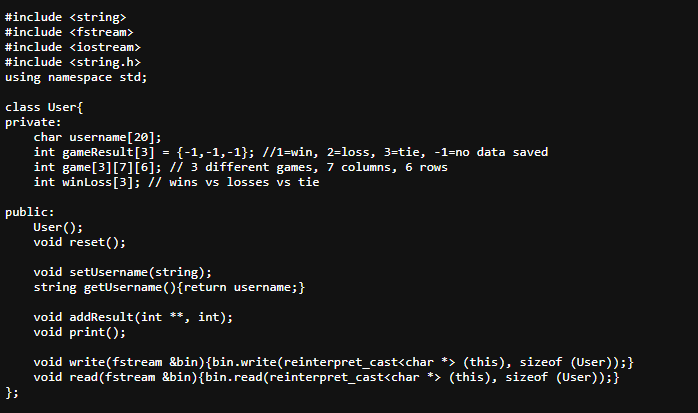
**Binary Interface:**

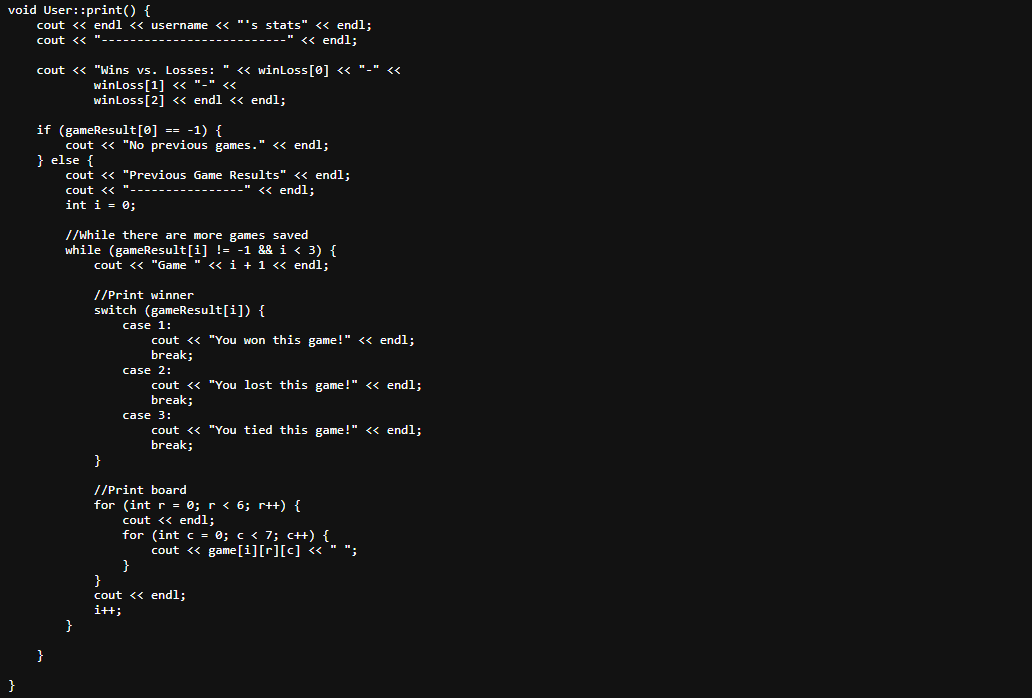
****

**More of Binary Interface class (updateRecord function)**

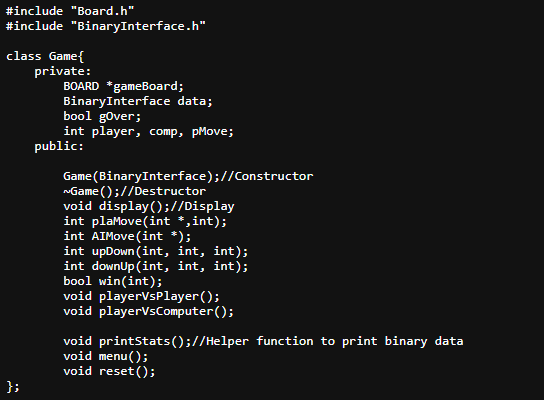
****

**User:**

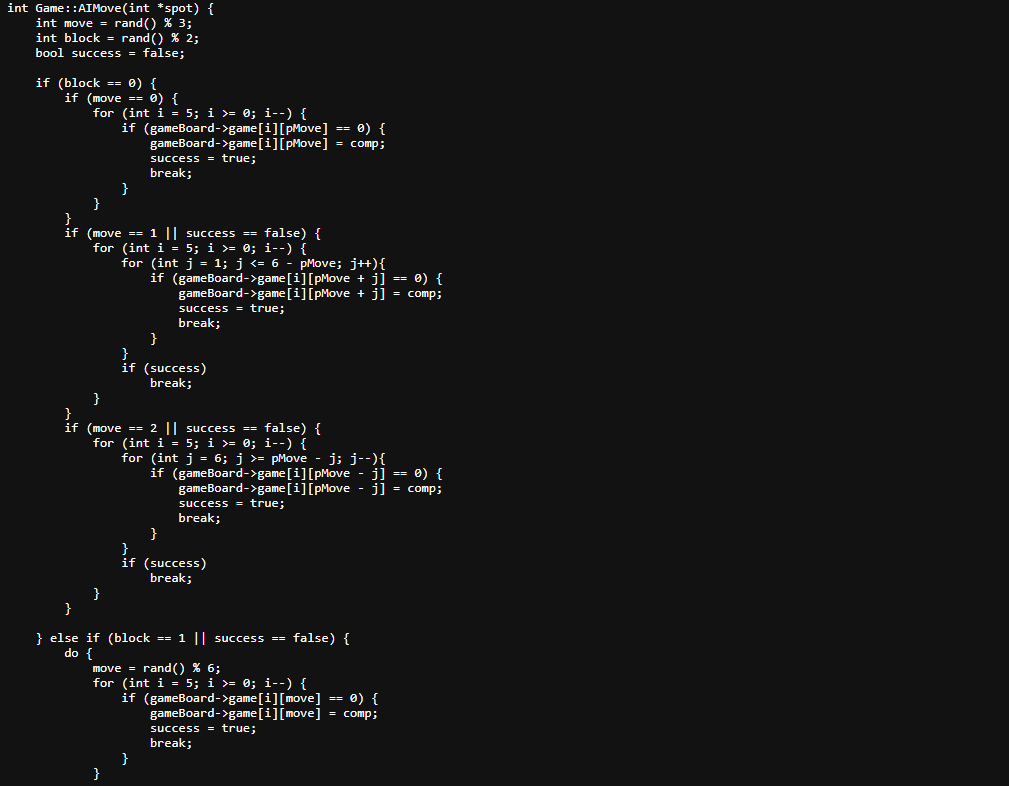
****

****

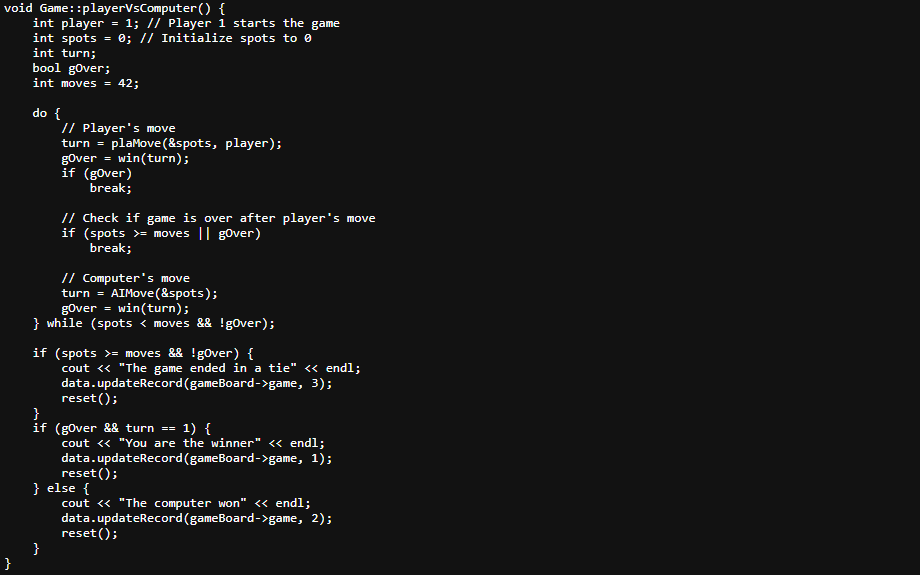
**Game:**

****

**More of Game class (AIMove Function)**

****

**More of Game class(PlaverVsComputer function)**

****

**9. Proof of working code**