Assignment Report

A Report for

EDE2102 Object Oriented Programming

Lecturer: Dr. Scott Jones

Lab: Dr. Oran

By:

Session: P2 Group: 6

Mohamed Nurandi Bin Mohamed Arman	2102018
Lim Jun Ting	2101406
Soon Duan Zhi Benedict	2101858
Zhao Ziyi	2102206

B.Eng. (Hons.) Electronics and Data Engineering

Singapore Institute of Technology
Technical University of Munich

Introduction

Football excitement surges as the 2022 World Cup is around the corner. The 2022 Qatar World Cup will be a historic event as it will be the first time a Word Cup is held in an Arab and Muslim-majority country this winter. Many football fans will be keeping up with the latest news and updates of World Cup 2022, which will excite most of the social media and application platforms. With such a huge audience in this upcoming 2022 World Cup, our team has developed an interactive sports game for people who like to bet on their favorite teams to win.

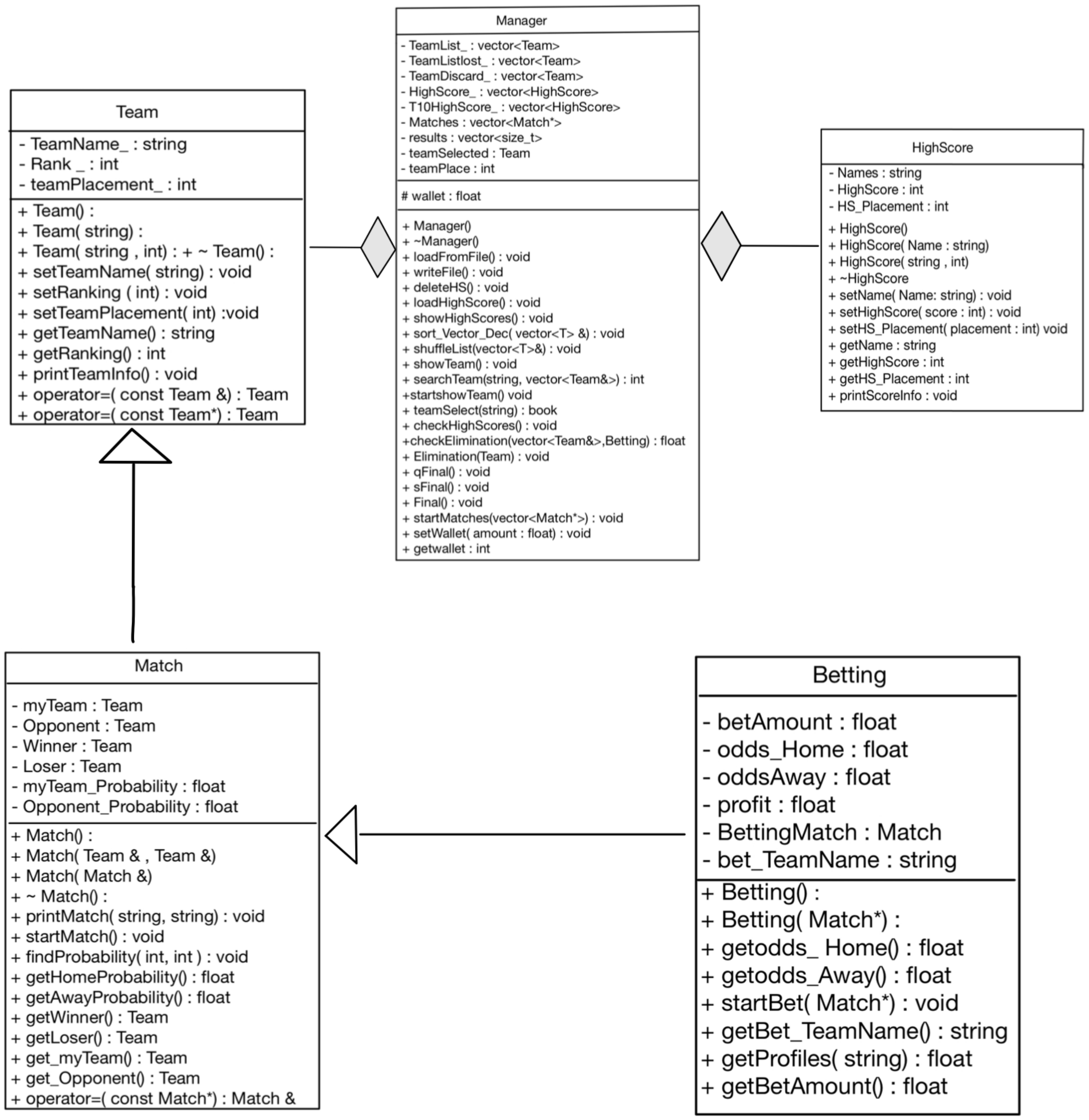
Description

Our team uses Replit and Visual Studio throughout the progression. We tried implementing GUI through Visual Studio, however, due to lack of time and experience, we were not able to fully produce a GUI in our game.
Agile methodology was used in our project because it was liable to changes along the way and more importantly we want to see a speedy progression rather than a perfect result. Along the way, we had to make sure our code met the requirements and optional objectives. Hence, at times, new ideas are often being implemented to suit the project requirements given. For example, we initially planned our project to be a tournament simulation. However, we decided we needed to include more user interaction and that is when we introduced a betting functionality.

UML Diagram

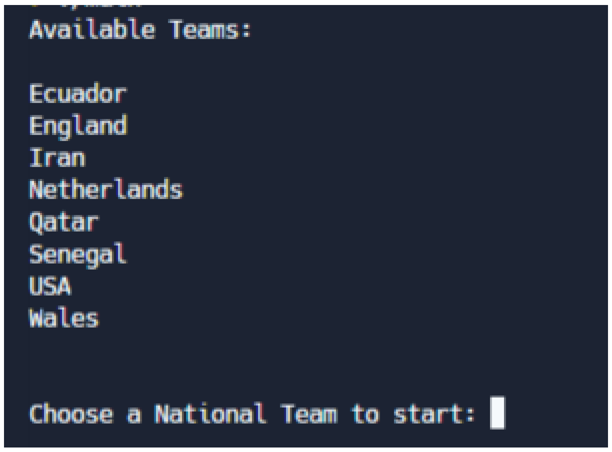
UML Class Diagram

(Manager Class Diagram)


Manager
- TeamList_ : vector<Team>
- TeamListlost_ : vector<Team>
- TeamDiscard_ : vector<Team>
- HighScore_ : vector<HighScore>
- T10HighScore_ : vector<HighScore>
- Matches : vector<Match*>
- results : vector<size_t>
- teamSelected : Team
- teamPlace : int
# wallet : float
+ Manager()
+ ~Manager()
+ loadFromFile() : void
+ writeFile() : void
+ deleteHS() : void
+ loadHighScore() : void
+ showHighScores() : void
+ sort_Vector_Dec(vector<T> &) : void
+ shuffleList(vector<T>&) : void
+ showTeam() : void
+ searchTeam(string, vector<Team&>) : int
+startshowTeam() void
+ teamSelect(string) : book
+ checkHighScores() : void
+checkElimination(vector<Team&>,Betting) : float
+ Elimination(Team) : void
+ qFinal() : void
+ sFinal() : void
+ Final() : void
+ startMatches(vector<Match*>) : void
+ setWallet(amount : float) : void
+ getwallet : int
Team
- TeamName_ : string
- Rank _ : int
- teamPlacement_ : int
+ Team() :
+ Team(string) :
+ Team(string , int) : + ~ Team() :
+ setTeamName(string) : void
+ setRanking (int) : void
+ setTeamPlacement(int) :void
+ getTeamName() : string
+ getRanking() : int
+ printTeamInfo() : void
+ operator=(const Team &) : Team
+ operator=(const Team*) : Team
HighScore
- Names : string
- HighScore : int
- HS_Placement : int
+ HighScore()
+ HighScore(Name : string)
+ HighScore(string , int)
+ ~HighScore
+ setName(Name: string) : void
+ setHighScore(score : int) : void
+ setHS_Placement(placement : int) void
+ getName : string
+ getHighScore : int
+ getHS_Placement : int
+ printScoreInfo : void
Match
- myTeam : Team
- Opponent : Team
- Winner : Team
- Loser : Team
- myTeam_Probability : float
- Opponent_Probability : float
+ Match() :
+ Match(Team & , Team &)
+ Match(Match &)
+ ~ Match() :
+ printMatch(string, string) : void
+ startMatch() : void
+ findProbability(int, int) : void
+ getHomeProbability() : float
+ getAwayProbability() : float
+ getWinner() : Team
+ getLoser() : Team
+ get_myTeam() : Team
+ get_Opponent() : Team
+ operator=(const Match*) : Match &
Betting
- betAmount : float
- odds_Home : float
- oddsAway : float
- profit : float
- BettingMatch : Match
- bet_TeamName : string
+ Betting() :
+ Betting(Match*) :
+ getodds_ Home() : float
+ getodds_Away() : float
+ startBet(Match*) : void
+ getBet_TeamName() : string
+ getProfiles(string) : float
+ getBetAmount() : float

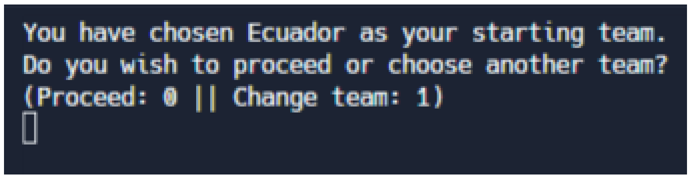
How to Play

1) Player will be tasked to choose a team of 8 and determine who is the final winner:



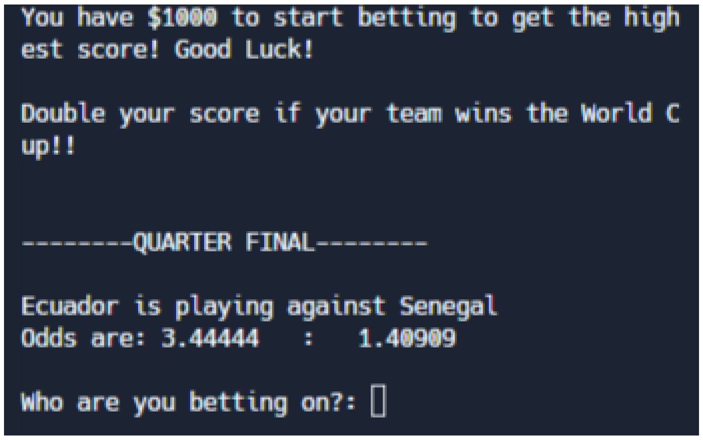
```
Available Teams:

Ecuador
England
Iran
Netherlands
Qatar
Senegal
USA
Wales


Choose a National Team to start:
```

2) After choosing their winning team, player will be given a choice to proceed or choose another team :



```
You have chosen Ecuador as your starting team.
Do you wish to proceed or choose another team?
(Proceed: 0 || Change team: 1)
```

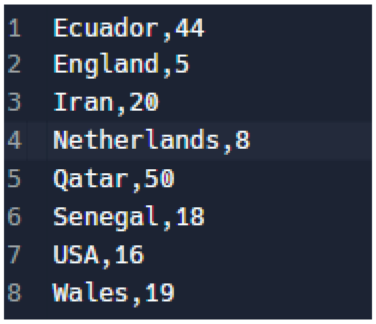
3) Once proceed :0, the player will start off with $1000 on the bet.
   Upon betting on the winning team in the finals, the player will receive double as an incentive.
   - The country that versus player is randomize on ShuffleList.
   - Player will be asked to bet on the winners with the odds given



```
You have $1000 to start betting to get the high
est score! Good Luck!

Double your score if your team wins the World C
up!!


--------QUARTER FINAL--------

Ecuador is playing against Senegal
Odds are: 3.44444   :   1.40909

Who are you betting on?: 
```

In the Quater Final:

- The player will be asked to choose who to bet on between two countries.
- The Odds, the algorithm chosen by our team is based on the ranking and probability of the matches.
- Ranking is based on our CSV file



```
Ecuador,44
England,5
Iran,20
Netherlands,8
Qatar,50
Senegal,18
USA,16
Wales,19
```

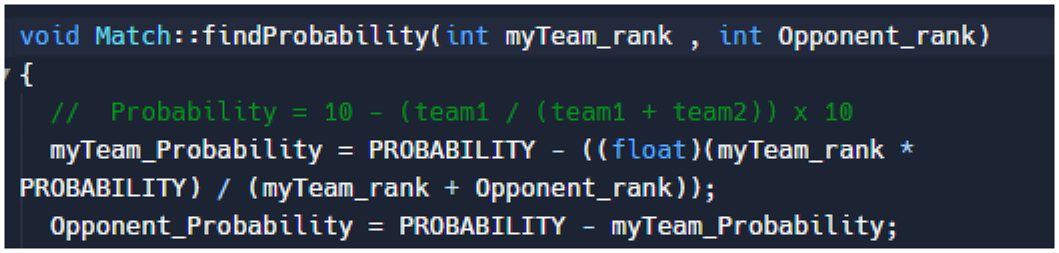
The mathematical calculation for probability between ranks:

$$P_1 = 10 - \left(\frac{rank_1}{rank_1 + rank_2} * 10\right)$$

P – Probability of team winning

$rank_1$ – Fifa ranking of team 1

$rank_2$ – Fifa ranking of team 2


```
void Match::findProbability(int myTeam_rank , int Opponent_rank)
{
  //  Probability = 10 - (team1 / (team1 + team2)) x 10
  myTeam_Probability = PROBABILITY - ((float)(myTeam_rank *
PROBABILITY) / (myTeam_rank + Opponent_rank));
  Opponent_Probability = PROBABILITY - myTeam_Probability;
```

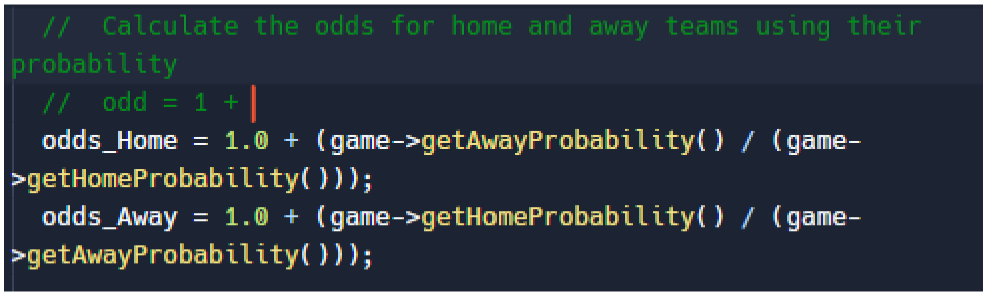
The Mathematical calculation for odds between Home and Away by their probability for winning:

$$odds_1 = 1 + (\frac{P_2}{P_1})$$

$odds$ – Amount that can be won

P_1 – Winning probability of team 1

P_2 – Winning probability of team 2


```
  //  Calculate the odds for home and away teams using their
probability
  //  odd = 1 +
  odds_Home = 1.0 + (game->getAwayProbability() / (game-
>getHomeProbability()));
  odds_Away = 1.0 + (game->getHomeProbability() / (game-
>getAwayProbability()));
```

Random_device was used as a pointer from 1-10 on the probability to determine who is the winner between two countries:

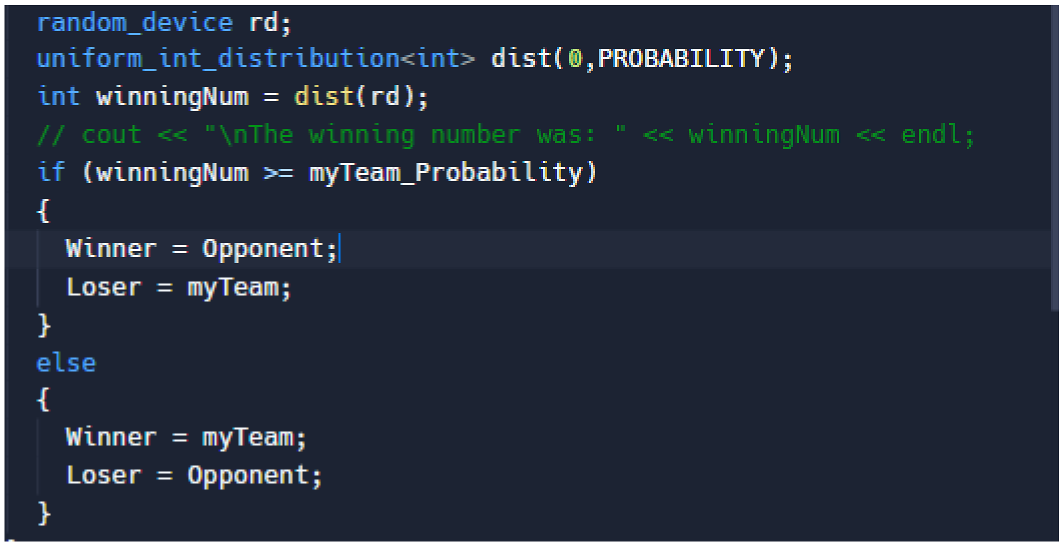
Example:
Netherlands VS USA – Probability of 7 : 3

Netherlands USA

|<-->||<------------------------->|

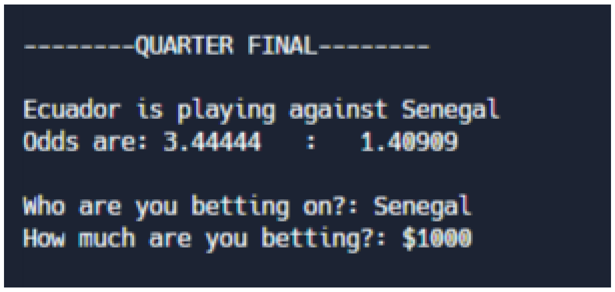
1	2	3	4	5	6	7	8	9	10

If the number generator outputs a number of 1-7, the Netherlands wins. However, if the number generator outputs a number of 8-10,the USA wins.

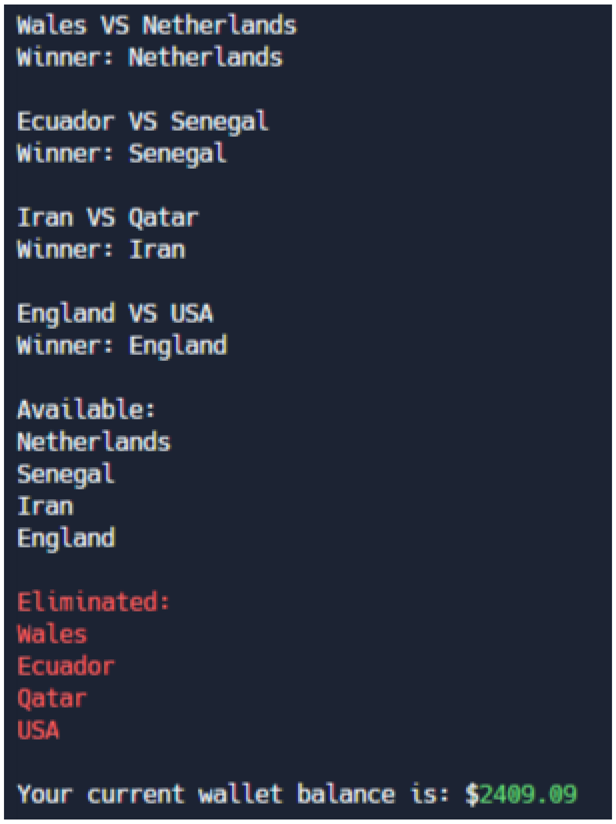


```
random_device rd;
uniform_int_distribution<int> dist(0,PROBABILITY);
int winningNum = dist(rd);
// cout << "\nThe winning number was: " << winningNum << endl;
if (winningNum >= myTeam_Probability)
{
  Winner = Opponent;
  Loser = myTeam;
}
else
{
  Winner = myTeam;
  Loser = Opponent;
}
```

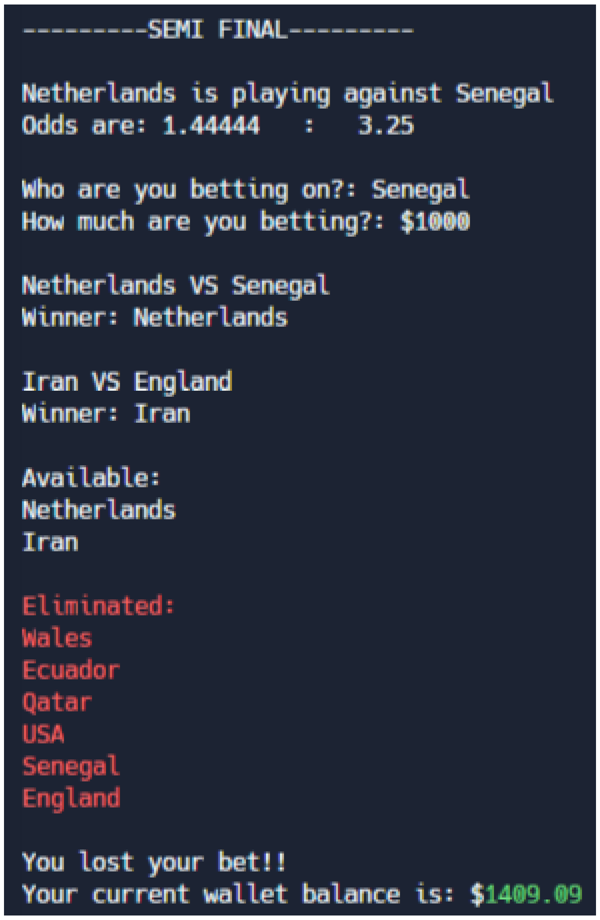
4) After betting the winning country, player will be inserting their bets $.


```
--------QUARTER FINAL--------

Ecuador is playing against Senegal
Odds are: 3.44444   :   1.40909

Who are you betting on?: Senegal
How much are you betting?: $1000
```

5) Results will be shown immediately. The player will be able to see if their betting team win/lose. After the Quarter Finals, 4 teams will be eliminated from the list and leaving only 4 winning teams. The winning teams will then proceed to the Semi-Finals. The 4 winning teams will then be randomized to versus each other. Players also will be able to see their current wallet balance as shown below:

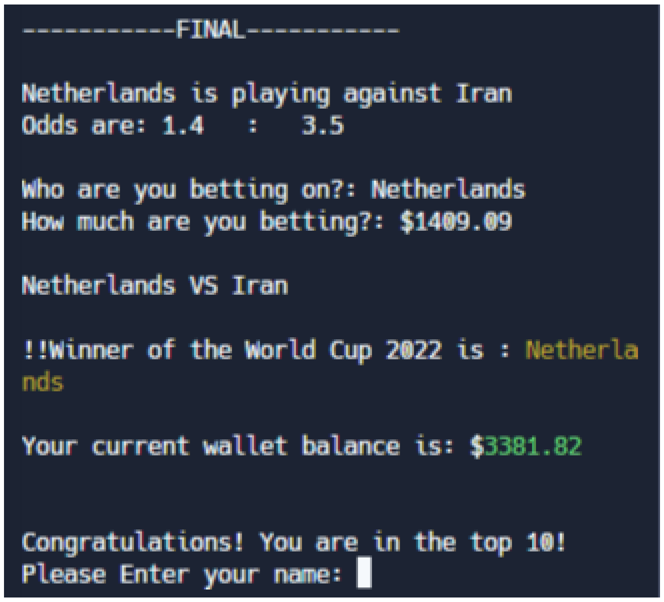

```
Wales VS Netherlands
Winner: Netherlands

Ecuador VS Senegal
Winner: Senegal

Iran VS Qatar
Winner: Iran

England VS USA
Winner: England

Available:
Netherlands
Senegal
Iran
England

Eliminated:
Wales
Ecuador
Qatar
USA

Your current wallet balance is: $2409.09
```

6) In the Semi-Final, the same queries and algorithm will be in place. In this scenario, if the final winning team that player's bet is lost. The game will continue to play on and allow players to bet on the winning team.

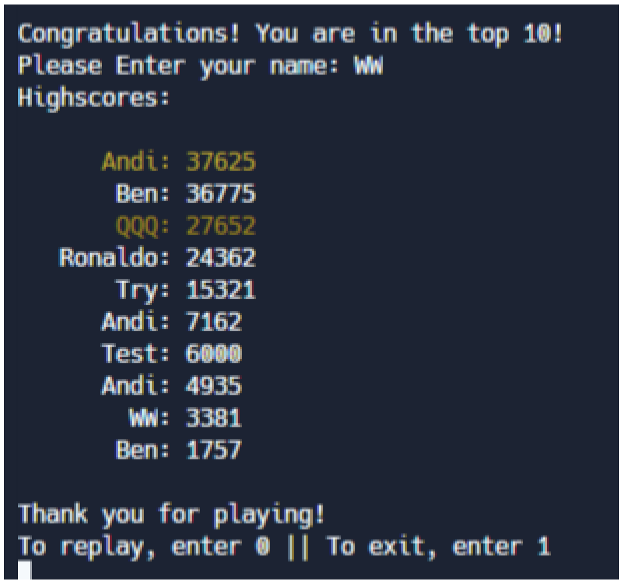


```
---------SEMI FINAL---------

Netherlands is playing against Senegal
Odds are: 1.44444   :   3.25

Who are you betting on?: Senegal
How much are you betting?: $1000

Netherlands VS Senegal
Winner: Netherlands

Iran VS England
Winner: Iran

Available:
Netherlands
Iran

Eliminated:
Wales
Ecuador
Qatar
USA
Senegal
England

You lost your bet!!
Your current wallet balance is: $1409.09
```

7) In the Final, the Game will announce the Winner of the World Cup 2022 after the final bet. The Player's final wallet balance will be shown and a congratulatory statement will be shown as well if it is listed in the top 10. Highscore board will be shown regardless of winning/losing at the finals and losing all bets at the beginning. The player will be able to insert their name if they are in the top 10 list.



```
-----------FINAL-----------

Netherlands is playing against Iran
Odds are: 1.4   :   3.5

Who are you betting on?: Netherlands
How much are you betting?: $1409.09

Netherlands VS Iran

!!Winner of the World Cup 2022 is : Netherla
nds

Your current wallet balance is: $3381.82


Congratulations! You are in the top 10!
Please Enter your name: 
```

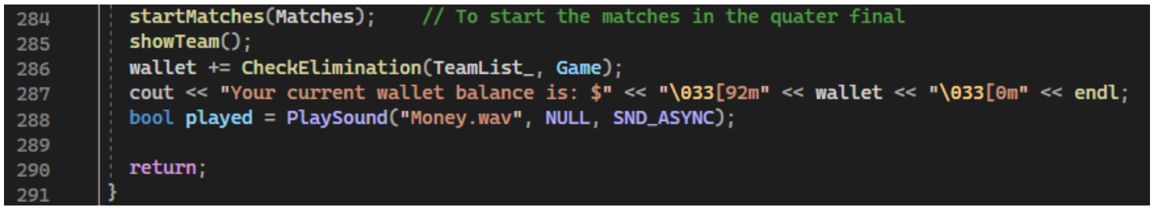


```
Congratulations! You are in the top 10!
Please Enter your name: WW
Highscores:

      Andi: 37625
       Ben: 36775
       QQQ: 27652
   Ronaldo: 24362
       Try: 15321
      Andi: 7162
      Test: 6000
      Andi: 4935
        WW: 3381
       Ben: 1757

Thank you for playing!
To replay, enter 0 || To exit, enter 1
```

Additional Features

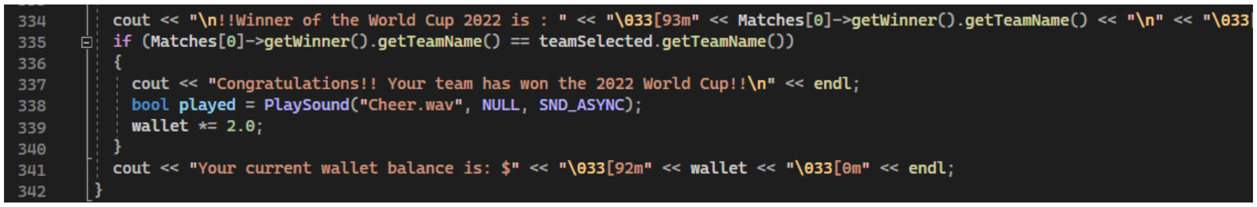
- **Sound effects**

  We have implemented some sound effects in our project by using the "Playsound" function.



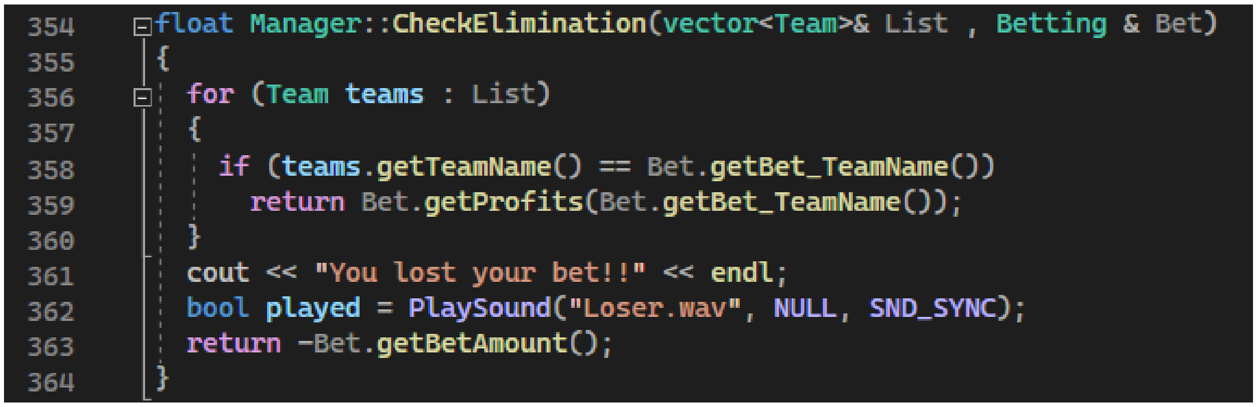
```cpp
    startMatches(Matches);    // To start the matches in the quater final
    showTeam();
    wallet += CheckElimination(TeamList_, Game);
    cout << "Your current wallet balance is: $" << "\033[92m" << wallet << "\033[0m" << endl;
    bool played = PlaySound("Money.wav", NULL, SND_ASYNC);

    return;
}
```

When wallet is being updated, Money.wav asynchronously.



```cpp
    cout << "\n!!Winner of the World Cup 2022 is : " << "\033[93m" << Matches[0]->getWinner().getTeamName() << "\n" << "\033
    if (Matches[0]->getWinner().getTeamName() == teamSelected.getTeamName())
    {
        cout << "Congratulations!! Your team has won the 2022 World Cup!!\n" << endl;
        bool played = PlaySound("Cheer.wav", NULL, SND_ASYNC);
        wallet *= 2.0;
    }
    cout << "Your current wallet balance is: $" << "\033[92m" << wallet << "\033[0m" << endl;
}
```

When player's team has won, Cheer.wav will play asynchronously.



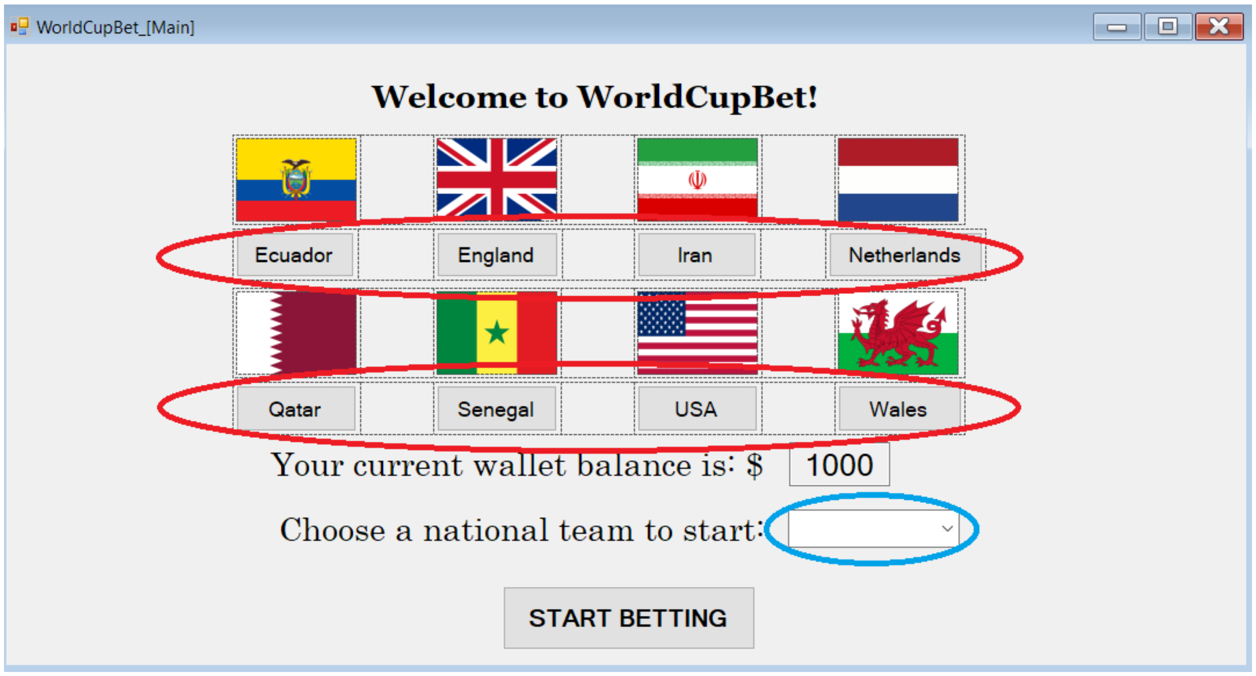
```cpp
float Manager::CheckElimination(vector<Team>& List , Betting & Bet)
{
    for (Team teams : List)
    {
        if (teams.getTeamName() == Bet.getBet_TeamName())
            return Bet.getProfits(Bet.getBet_TeamName());
    }
    cout << "You lost your bet!!" << endl;
    bool played = PlaySound("Loser.wav", NULL, SND_SYNC);
    return -Bet.getBetAmount();
}
```

When player has lost his/her bet, Loser.wav will play synchronously.

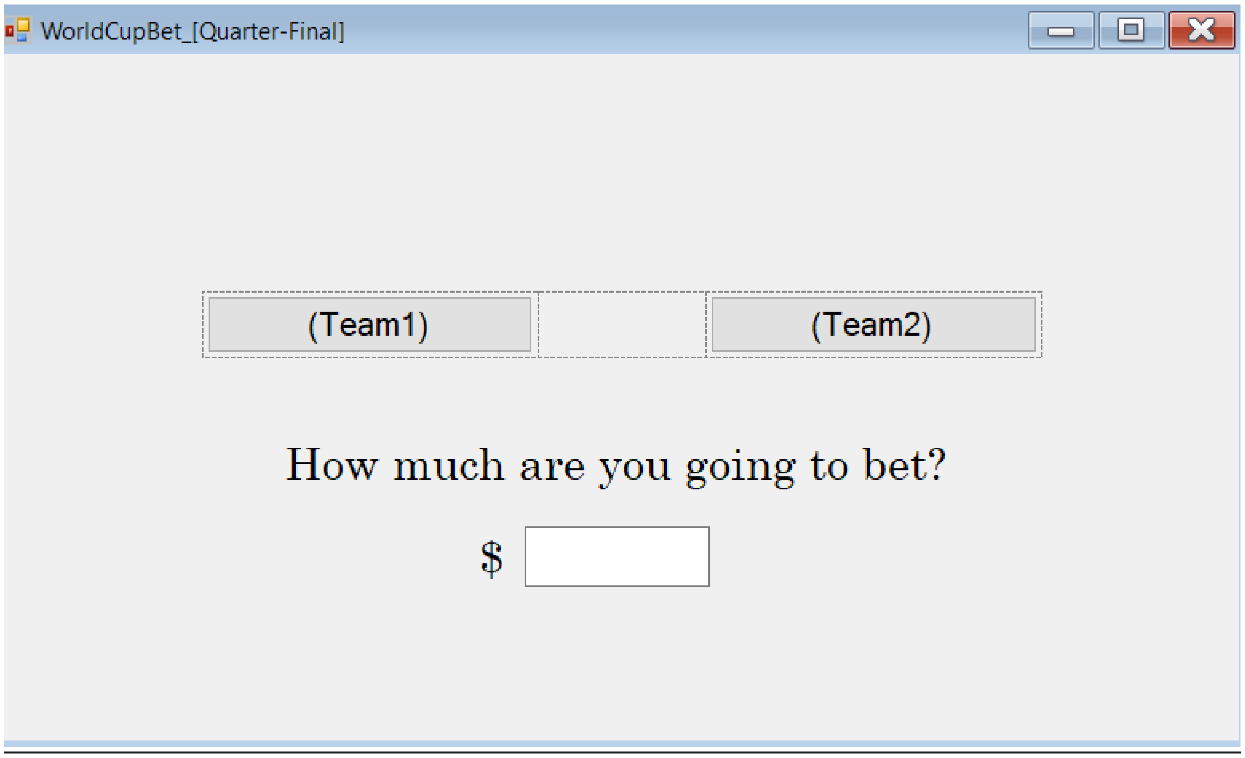
- **<u>GUI</u>**

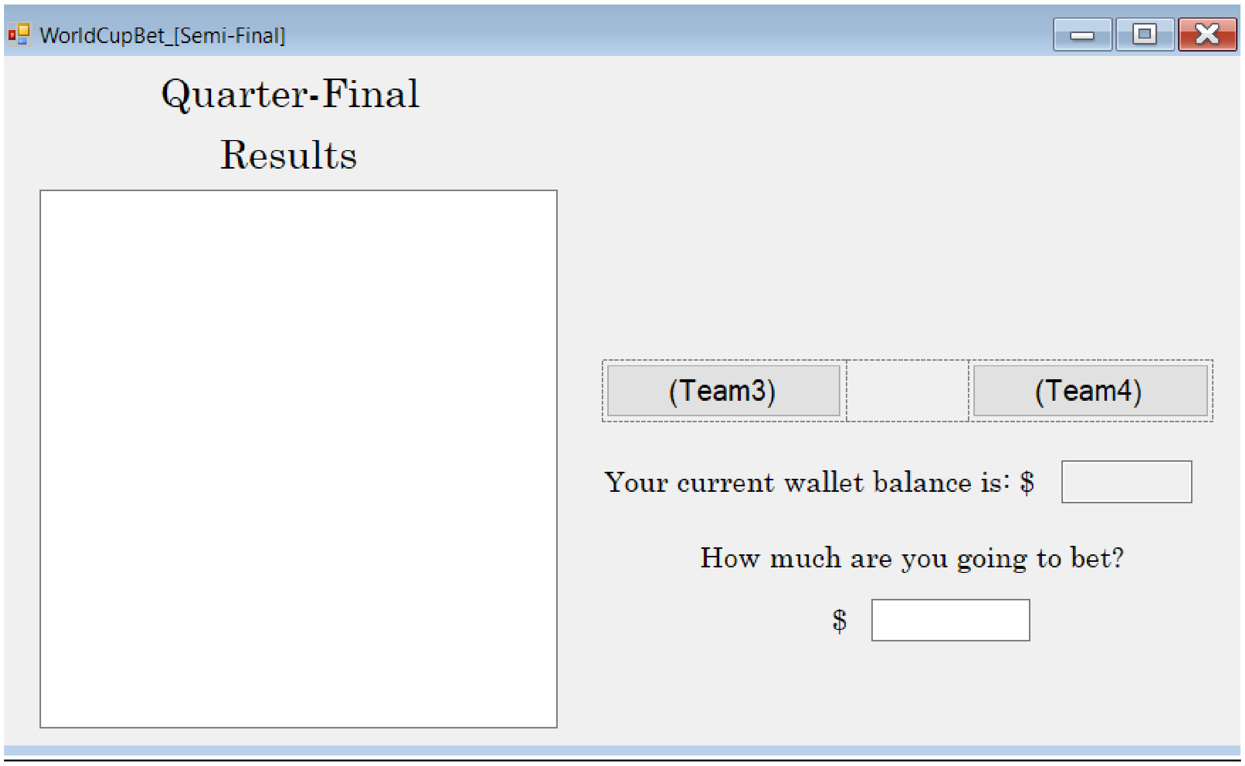
  In this project, we have attempted to implement GUI through Windows Forms but it was unsuccessful. However, we managed to create its structure.

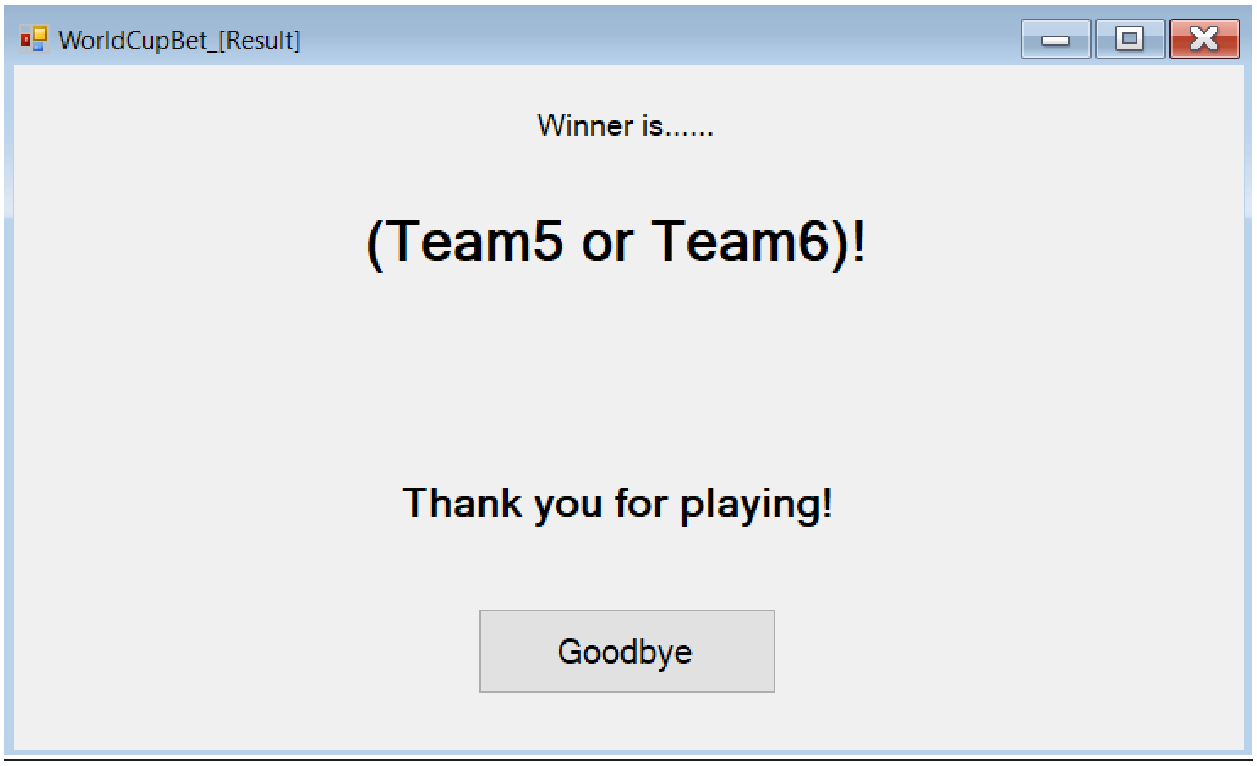
  8 teams and their corresponding flags will be shown on the main page. When one of the team's buttons (circled in red) is clicked, their corresponding Wikipedia page will pop-up, showing their national football team's information. Users can select a team from the drop-down menu (circled in blue) and proceed to the next stage by clicking "START BETTING".



We have constructed GUI from Quarter-Final to Final Results (shown below) without implementation of functions.


WorldCupBet_[Quarter-Final]
(Team1)
(Team2)
How much are you going to bet?
$


WorldCupBet_[Semi-Final]
Quarter-Final
Results
(Team3)
(Team4)
Your current wallet balance is: $
How much are you going to bet?
$



Winner is......

(Team5 or Team6)!

Thank you for playing!

<u>Contributions</u>

Mohamed Nurandi Bin Mohamed Arman

- Betting.cpp & Betting.h (100%)
- Match.cpp & Match.h (100%)
- Manager.cpp & Manager.h
    - int Manager::searchTeam(string name, vector<Team>& List) (100%)
    - bool Manager::teamSelect(string name) (100%)
    - void Manager::qFinal() (100%)
    - Betting Game(Matches[bet]) (100%)
    - void Manager::sFinal() (100%)
    - void Manager::Final() (100%)
    - void Manager::startMatches(vector<Match*> Games) (100%)
    - float Manager::CheckElimination(vector<Team>& List , Betting & Bet) (100%)
    - void Manager::Eliminate(Team loser) (100%)
    - void Manager::sort_Vector_Dec(std::vector<T> & List) (100%)
    - void Manager::shuffleList(vector<T> & List) (100%)
    - int getRandomNumber(int min , int max) (100%)

Lim Jun Ting

- Main.cpp file (100%)
- UML (100%)
- Report (100%)
- Teams.cpp & Teams.h (100%)

Soon Duan Zhi Benedict

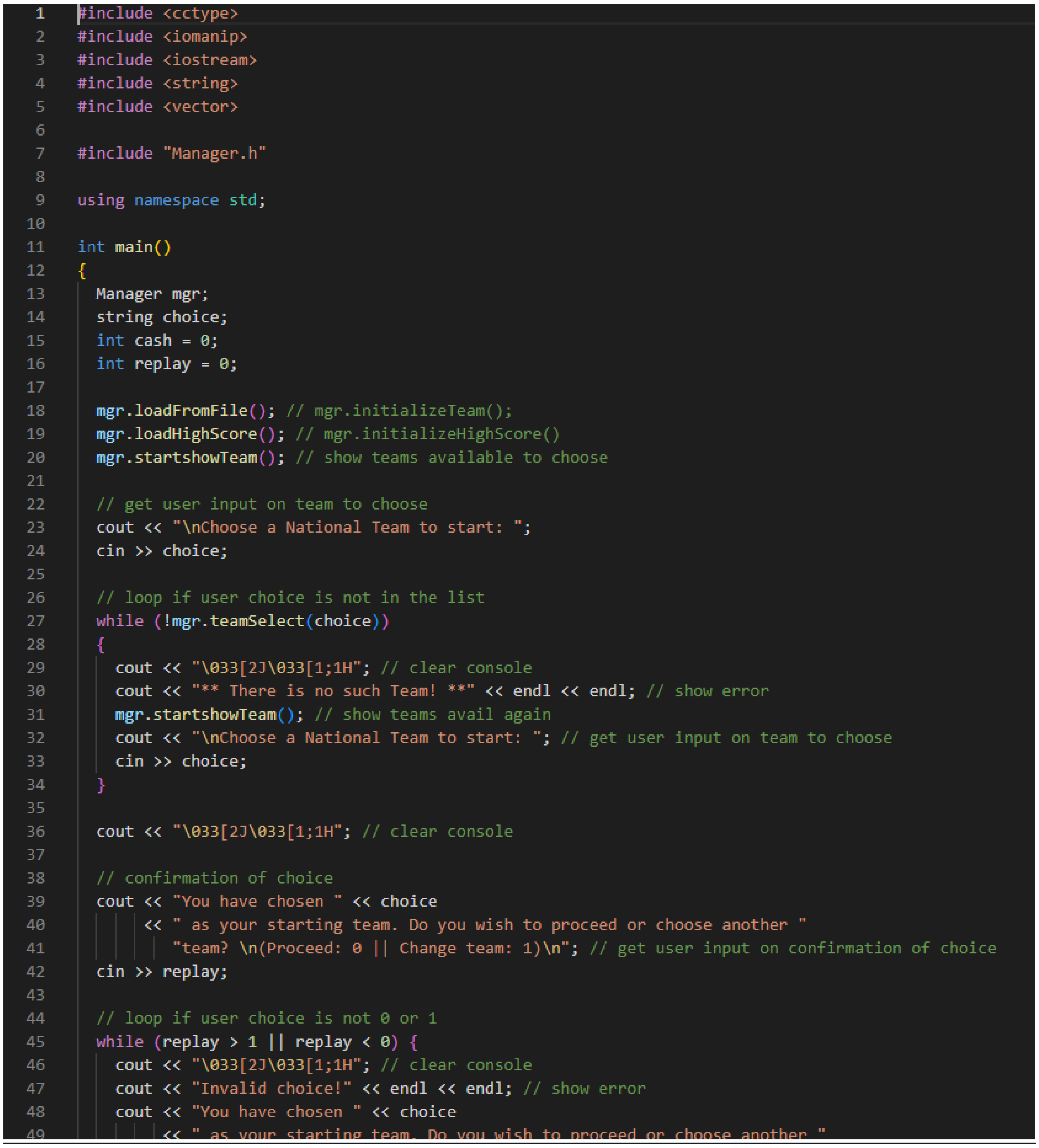
- Manager.cpp & Manager.h
    - void Manager::loadFromFile() (100%)
    - void Manager::loadHighScore() (100%)
    - void Manager::writeFile() (100%)
    - void Manager::deleteHS() (100%)
    - void Manager::checkHighScores() (100%)
    - void Manager::showHighScores() (100%)
    - void Manager::startshowTeam() (100%)
    - void Manager::showTeam() (100%)
- Highscore.cpp & Highscore.h (100%)
- Teams.cpp & Teams.h (100%)

Zhao Ziyi

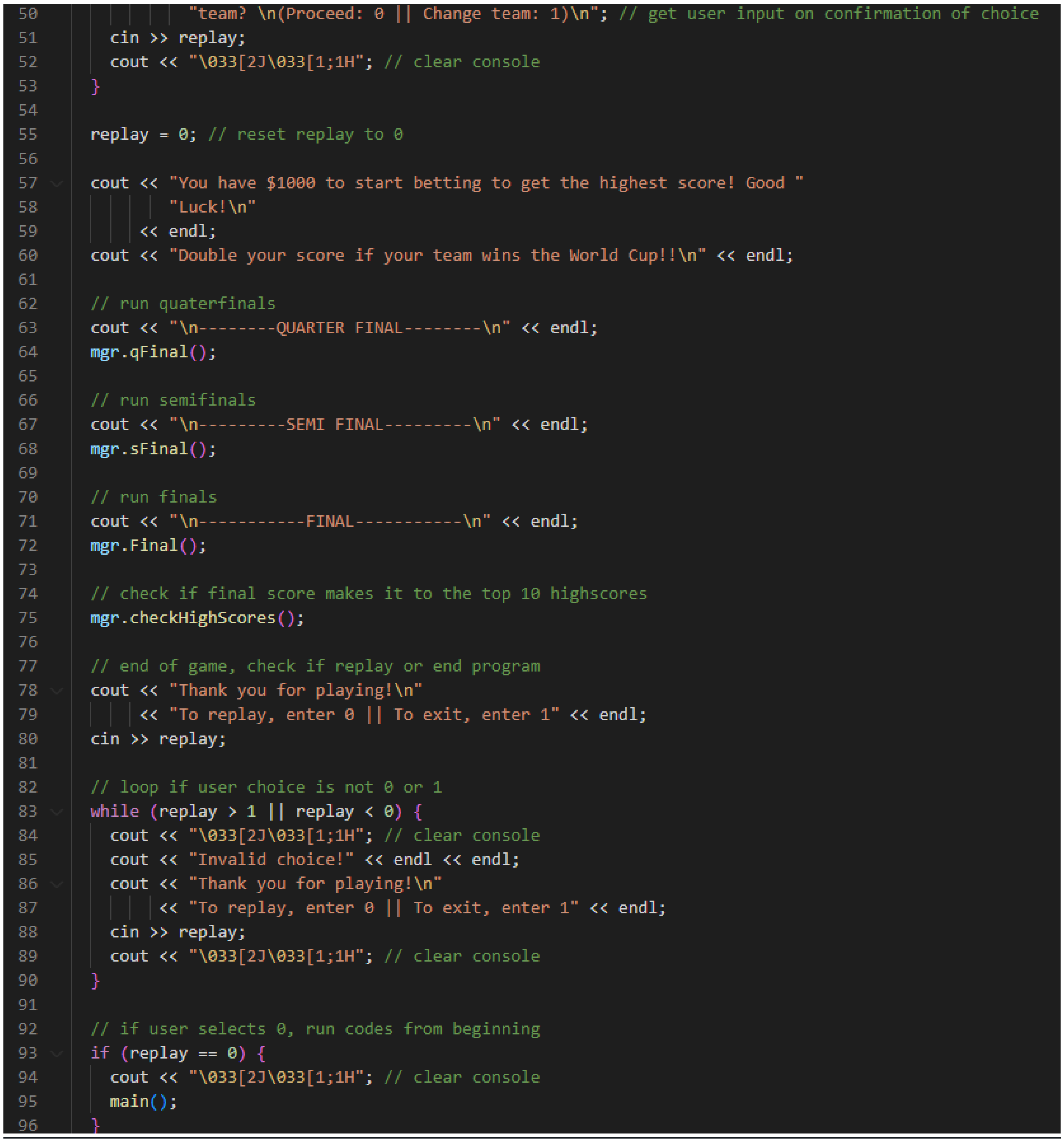
- GUI Implementation (50% completion)
    - Main.cpp
    - Main.h
    - Window_Quar.h
    - Window_Semi.h
    - Window_Final.h
    - Window_Result.h
- Sound Effects

Appendix

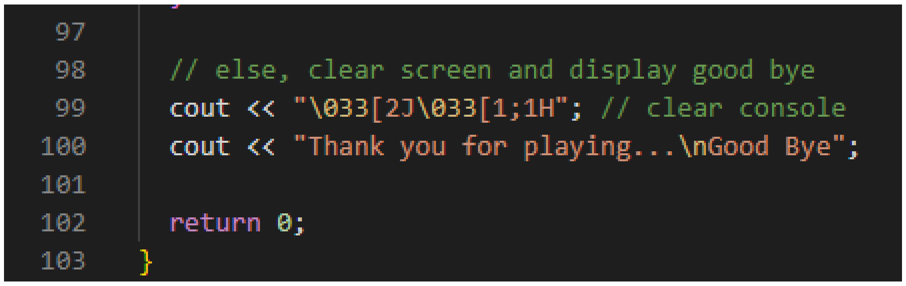
Main.cpp



```cpp
#include <cctype>
#include <iomanip>
#include <iostream>
#include <string>
#include <vector>

#include "Manager.h"

using namespace std;

int main()
{
  Manager mgr;
  string choice;
  int cash = 0;
  int replay = 0;

  mgr.loadFromFile(); // mgr.initializeTeam();
  mgr.loadHighScore(); // mgr.initializeHighScore()
  mgr.startshowTeam(); // show teams available to choose

  // get user input on team to choose
  cout << "\nChoose a National Team to start: ";
  cin >> choice;

  // loop if user choice is not in the list
  while (!mgr.teamSelect(choice))
  {
    cout << "\033[2J\033[1;1H"; // clear console
    cout << "** There is no such Team! **" << endl << endl; // show error
    mgr.startshowTeam(); // show teams avail again
    cout << "\nChoose a National Team to start: "; // get user input on team to choose
    cin >> choice;
  }

  cout << "\033[2J\033[1;1H"; // clear console

  // confirmation of choice
  cout << "You have chosen " << choice
       << " as your starting team. Do you wish to proceed or choose another "
          "team? \n(Proceed: 0 || Change team: 1)\n"; // get user input on confirmation of choice
  cin >> replay;

  // loop if user choice is not 0 or 1
  while (replay > 1 || replay < 0) {
    cout << "\033[2J\033[1;1H"; // clear console
    cout << "Invalid choice!" << endl << endl; // show error
    cout << "You have chosen " << choice
         << " as your starting team. Do you wish to proceed or choose another "
```

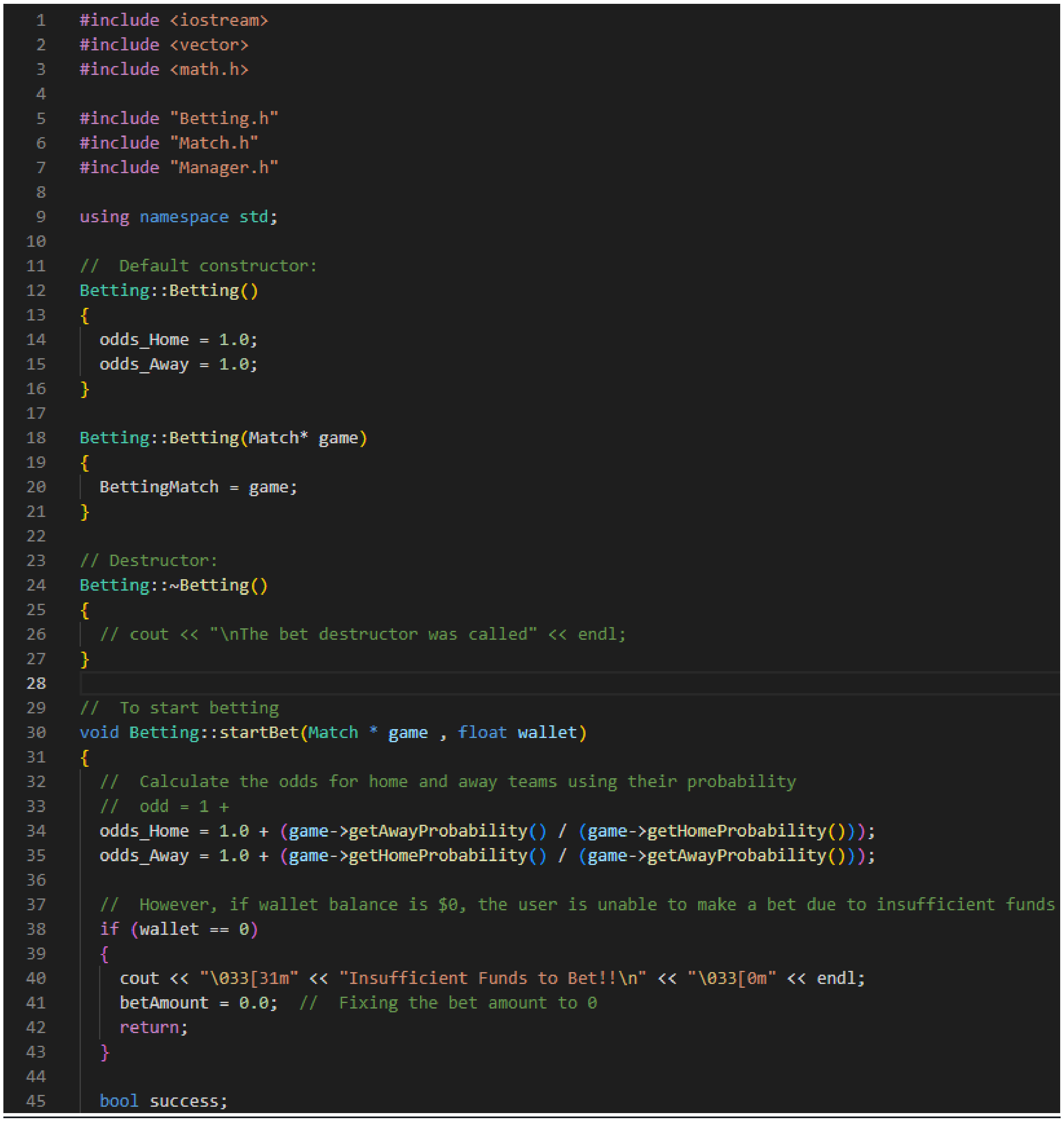


```cpp
            "team? \n(Proceed: 0 || Change team: 1)\n"; // get user input on confirmation of choice
        cin >> replay;
        cout << "\033[2J\033[1;1H"; // clear console
    }

    replay = 0; // reset replay to 0

    cout << "You have $1000 to start betting to get the highest score! Good "
            "Luck!\n"
         << endl;
    cout << "Double your score if your team wins the World Cup!!\n" << endl;

    // run quaterfinals
    cout << "\n--------QUARTER FINAL--------\n" << endl;
    mgr.qFinal();

    // run semifinals
    cout << "\n---------SEMI FINAL---------\n" << endl;
    mgr.sFinal();

    // run finals
    cout << "\n-----------FINAL-----------\n" << endl;
    mgr.Final();

    // check if final score makes it to the top 10 highscores
    mgr.checkHighScores();

    // end of game, check if replay or end program
    cout << "Thank you for playing!\n"
         << "To replay, enter 0 || To exit, enter 1" << endl;
    cin >> replay;

    // loop if user choice is not 0 or 1
    while (replay > 1 || replay < 0) {
        cout << "\033[2J\033[1;1H"; // clear console
        cout << "Invalid choice!" << endl << endl;
        cout << "Thank you for playing!\n"
             << "To replay, enter 0 || To exit, enter 1" << endl;
        cin >> replay;
        cout << "\033[2J\033[1;1H"; // clear console
    }

    // if user selects 0, run codes from beginning
    if (replay == 0) {
        cout << "\033[2J\033[1;1H"; // clear console
        main();
    }
```

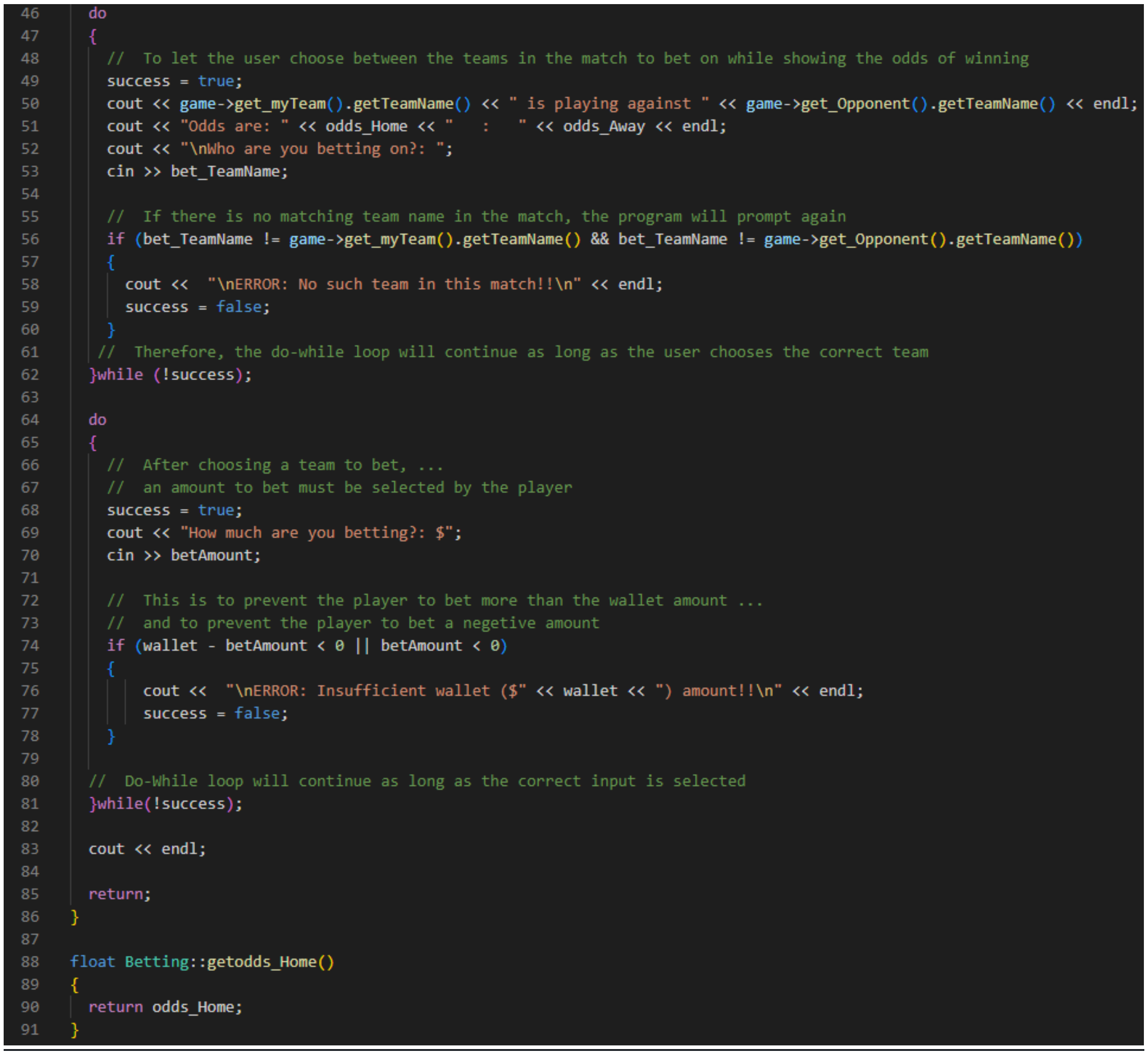


```cpp

    // else, clear screen and display good bye
    cout << "\033[2J\033[1;1H"; // clear console
    cout << "Thank you for playing...\nGood Bye";

    return 0;
}
```

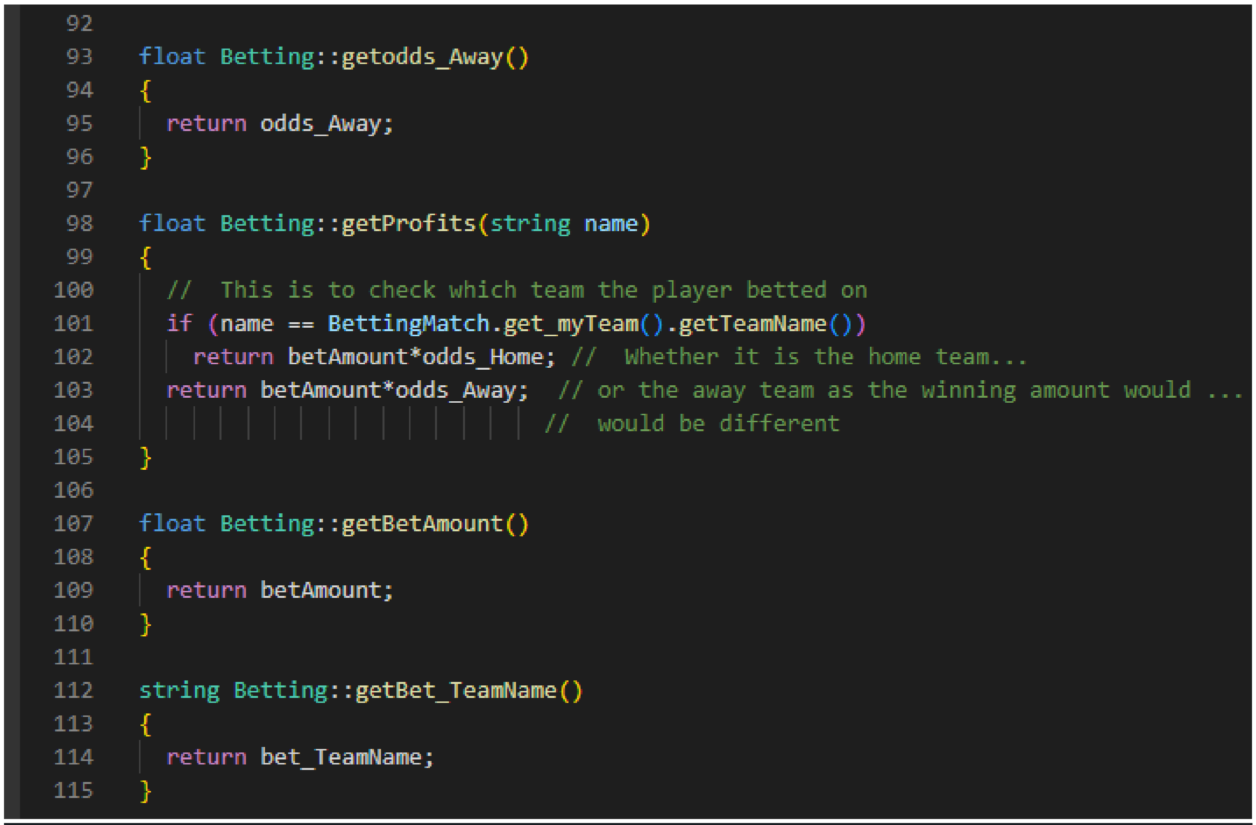
Betting.cpp



```cpp
#include <iostream>
#include <vector>
#include <math.h>

#include "Betting.h"
#include "Match.h"
#include "Manager.h"

using namespace std;

//  Default constructor:
Betting::Betting()
{
  odds_Home = 1.0;
  odds_Away = 1.0;
}

Betting::Betting(Match* game)
{
  BettingMatch = game;
}

// Destructor:
Betting::~Betting()
{
  // cout << "\nThe bet destructor was called" << endl;
}

//  To start betting
void Betting::startBet(Match * game , float wallet)
{
  //  Calculate the odds for home and away teams using their probability
  //  odd = 1 +
  odds_Home = 1.0 + (game->getAwayProbability() / (game->getHomeProbability()));
  odds_Away = 1.0 + (game->getHomeProbability() / (game->getAwayProbability()));

  //  However, if wallet balance is $0, the user is unable to make a bet due to insufficient funds
  if (wallet == 0)
  {
    cout << "\033[31m" << "Insufficient Funds to Bet!!\n" << "\033[0m" << endl;
    betAmount = 0.0;  //  Fixing the bet amount to 0
    return;
  }

  bool success;
```

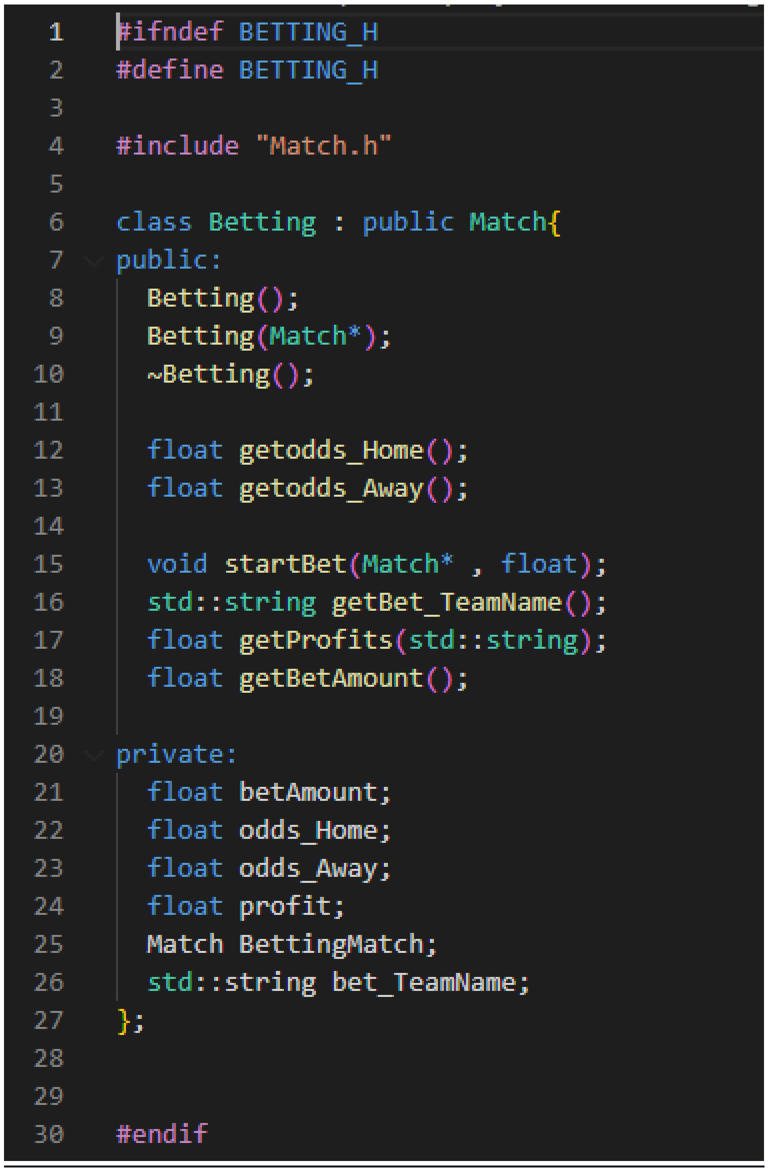


```
    do
    {
      //  To let the user choose between the teams in the match to bet on while showing the odds of winning
      success = true;
      cout << game->get_myTeam().getTeamName() << " is playing against " << game->get_Opponent().getTeamName() << endl;
      cout << "Odds are: " << odds_Home << "   :   " << odds_Away << endl;
      cout << "\nWho are you betting on?: ";
      cin >> bet_TeamName;

      //  If there is no matching team name in the match, the program will prompt again
      if (bet_TeamName != game->get_myTeam().getTeamName() && bet_TeamName != game->get_Opponent().getTeamName())
      {
        cout <<  "\nERROR: No such team in this match!!\n" << endl;
        success = false;
      }
     //  Therefore, the do-while loop will continue as long as the user chooses the correct team
    }while (!success);

    do
    {
      //  After choosing a team to bet, ...
      //  an amount to bet must be selected by the player
      success = true;
      cout << "How much are you betting?: $";
      cin >> betAmount;

      //  This is to prevent the player to bet more than the wallet amount ...
      //  and to prevent the player to bet a negetive amount
      if (wallet - betAmount < 0 || betAmount < 0)
      {
          cout <<  "\nERROR: Insufficient wallet ($" << wallet << ") amount!!\n" << endl;
          success = false;
      }

    //  Do-While loop will continue as long as the correct input is selected
    }while(!success);

    cout << endl;

    return;
  }

  float Betting::getodds_Home()
  {
    return odds_Home;
  }
```



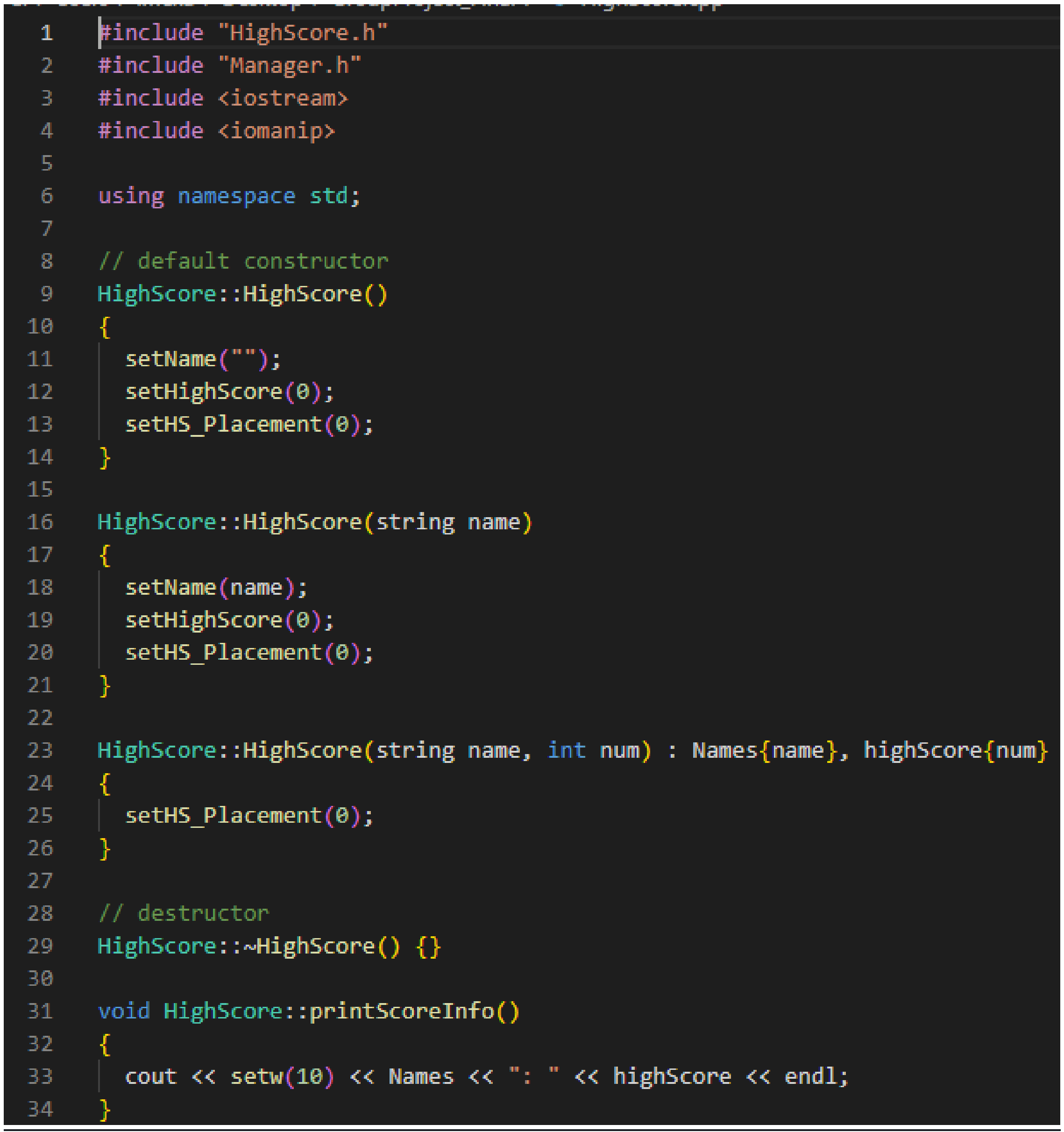
```

float Betting::getodds_Away()
{
    return odds_Away;
}

float Betting::getProfits(string name)
{
    //  This is to check which team the player betted on
    if (name == BettingMatch.get_myTeam().getTeamName())
        return betAmount*odds_Home; //  Whether it is the home team...
    return betAmount*odds_Away;  // or the away team as the winning amount would ...
                                //  would be different
}

float Betting::getBetAmount()
{
    return betAmount;
}

string Betting::getBet_TeamName()
{
    return bet_TeamName;
}
```

Betting.h



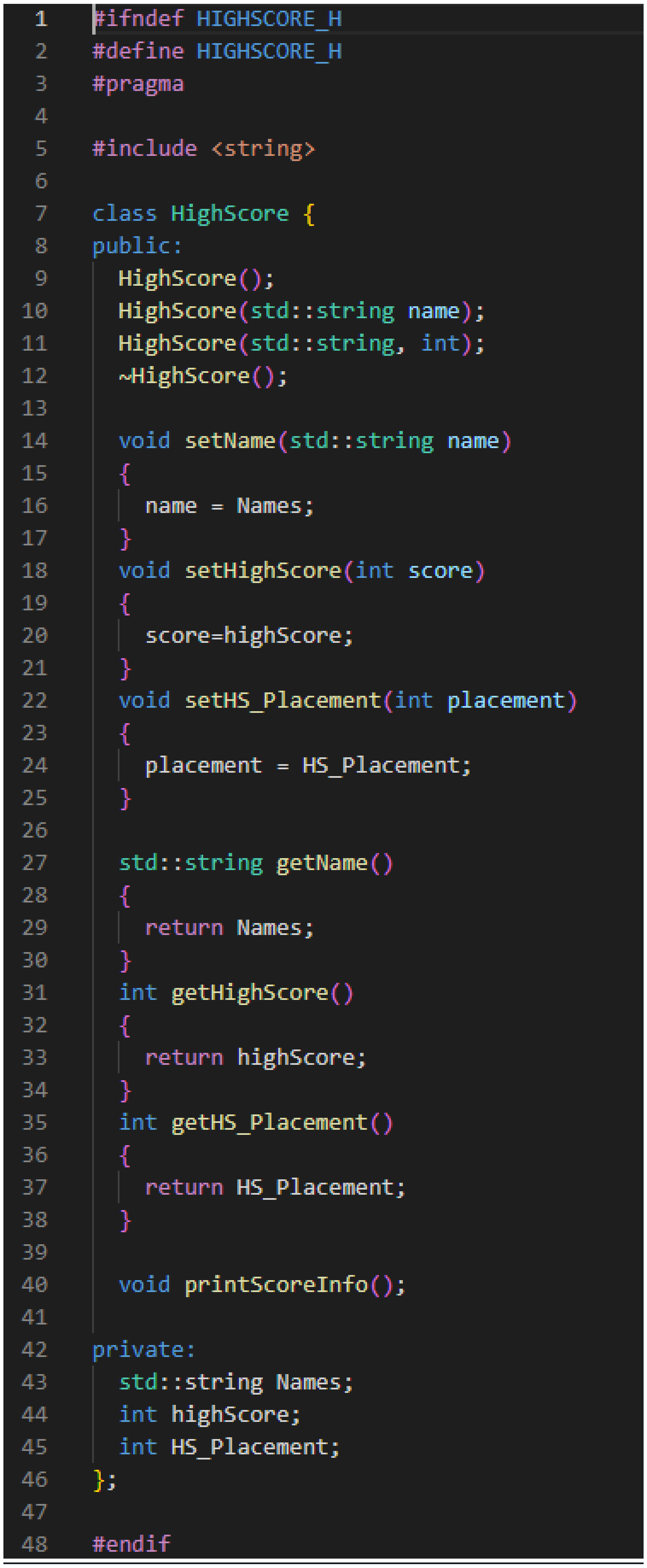
```cpp
#ifndef BETTING_H
#define BETTING_H

#include "Match.h"

class Betting : public Match{
public:
    Betting();
    Betting(Match*);
    ~Betting();

    float getodds_Home();
    float getodds_Away();

    void startBet(Match* , float);
    std::string getBet_TeamName();
    float getProfits(std::string);
    float getBetAmount();

private:
    float betAmount;
    float odds_Home;
    float odds_Away;
    float profit;
    Match BettingMatch;
    std::string bet_TeamName;
};


#endif
```

HighScore.cpp



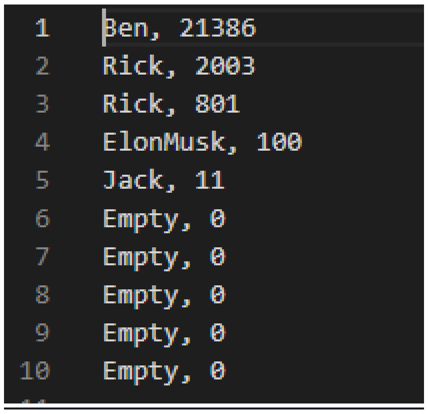
```cpp
#include "HighScore.h"
#include "Manager.h"
#include <iostream>
#include <iomanip>

using namespace std;

// default constructor
HighScore::HighScore()
{
    setName("");
    setHighScore(0);
    setHS_Placement(0);
}

HighScore::HighScore(string name)
{
    setName(name);
    setHighScore(0);
    setHS_Placement(0);
}

HighScore::HighScore(string name, int num) : Names{name}, highScore{num}
{
    setHS_Placement(0);
}

// destructor
HighScore::~HighScore() {}

void HighScore::printScoreInfo()
{
    cout << setw(10) << Names << ": " << highScore << endl;
}
```

HighScore.h



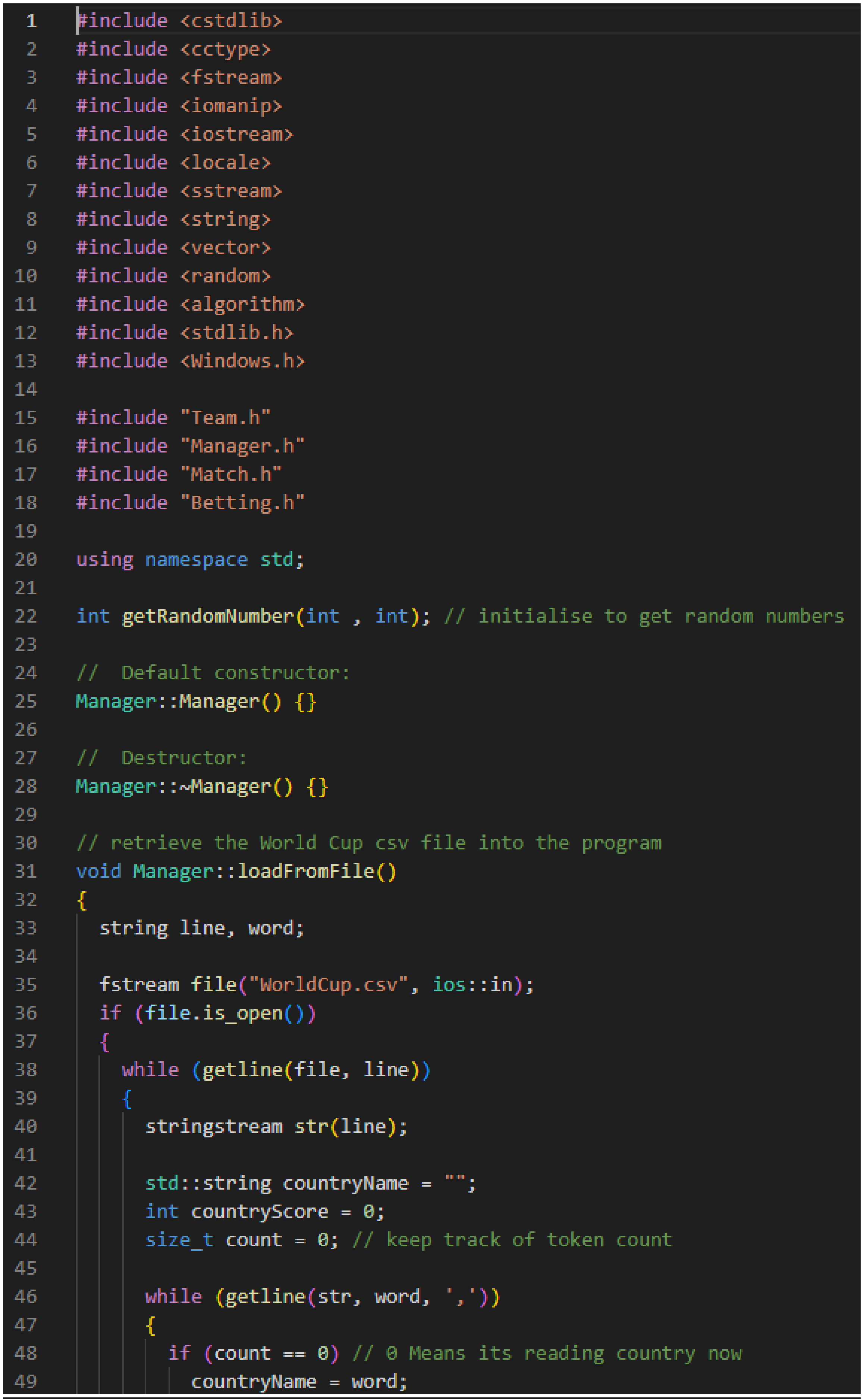
```cpp
#ifndef HIGHSCORE_H
#define HIGHSCORE_H
#pragma

#include <string>

class HighScore {
public:
  HighScore();
  HighScore(std::string name);
  HighScore(std::string, int);
  ~HighScore();

  void setName(std::string name)
  {
    name = Names;
  }
  void setHighScore(int score)
  {
    score=highScore;
  }
  void setHS_Placement(int placement)
  {
    placement = HS_Placement;
  }

  std::string getName()
  {
    return Names;
  }
  int getHighScore()
  {
    return highScore;
  }
  int getHS_Placement()
  {
    return HS_Placement;
  }

  void printScoreInfo();

private:
  std::string Names;
  int highScore;
  int HS_Placement;
};

#endif
```

HighScore.csv

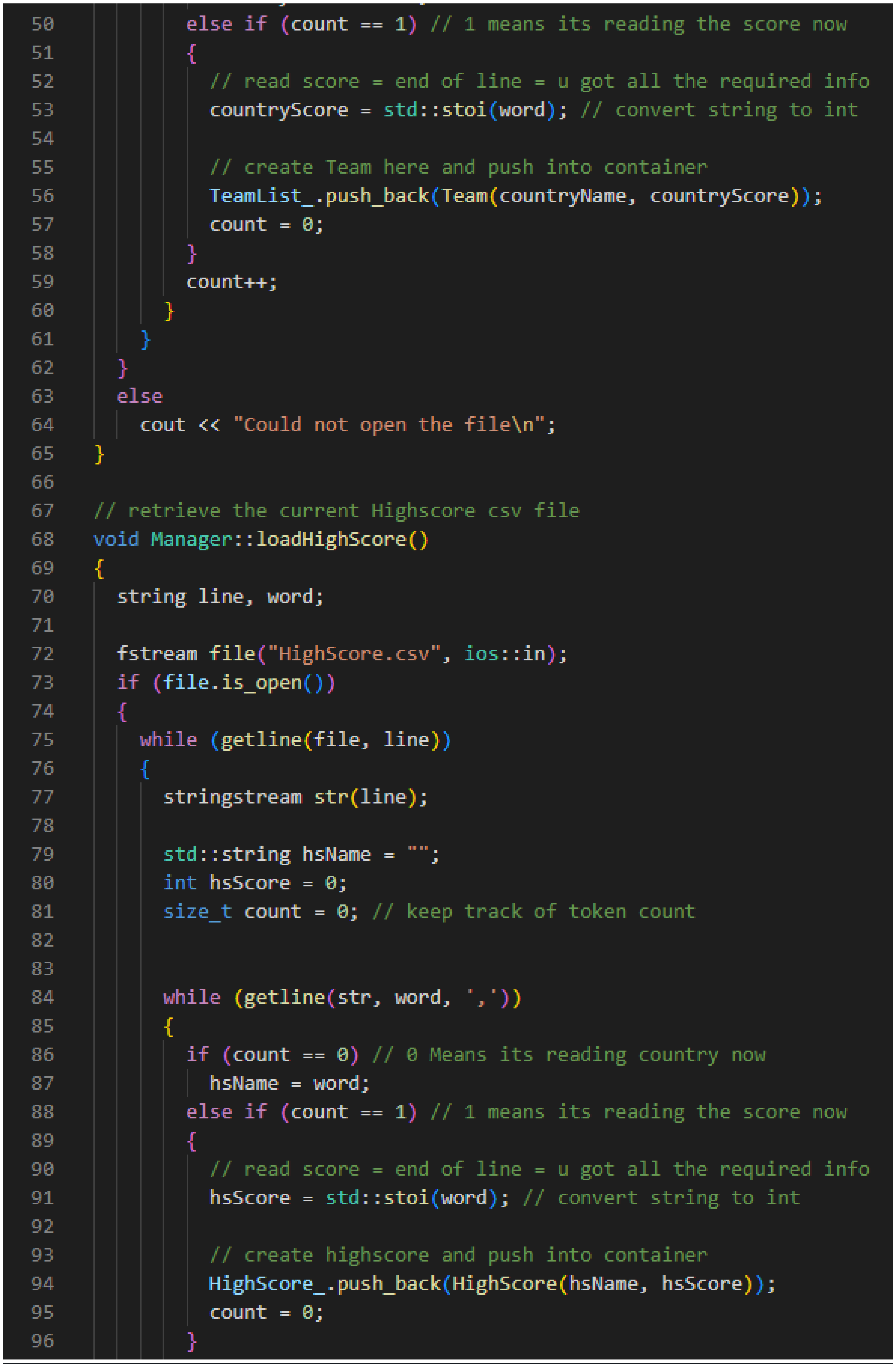


```
Ben, 21386
Rick, 2003
Rick, 801
ElonMusk, 100
Jack, 11
Empty, 0
Empty, 0
Empty, 0
Empty, 0
Empty, 0
```

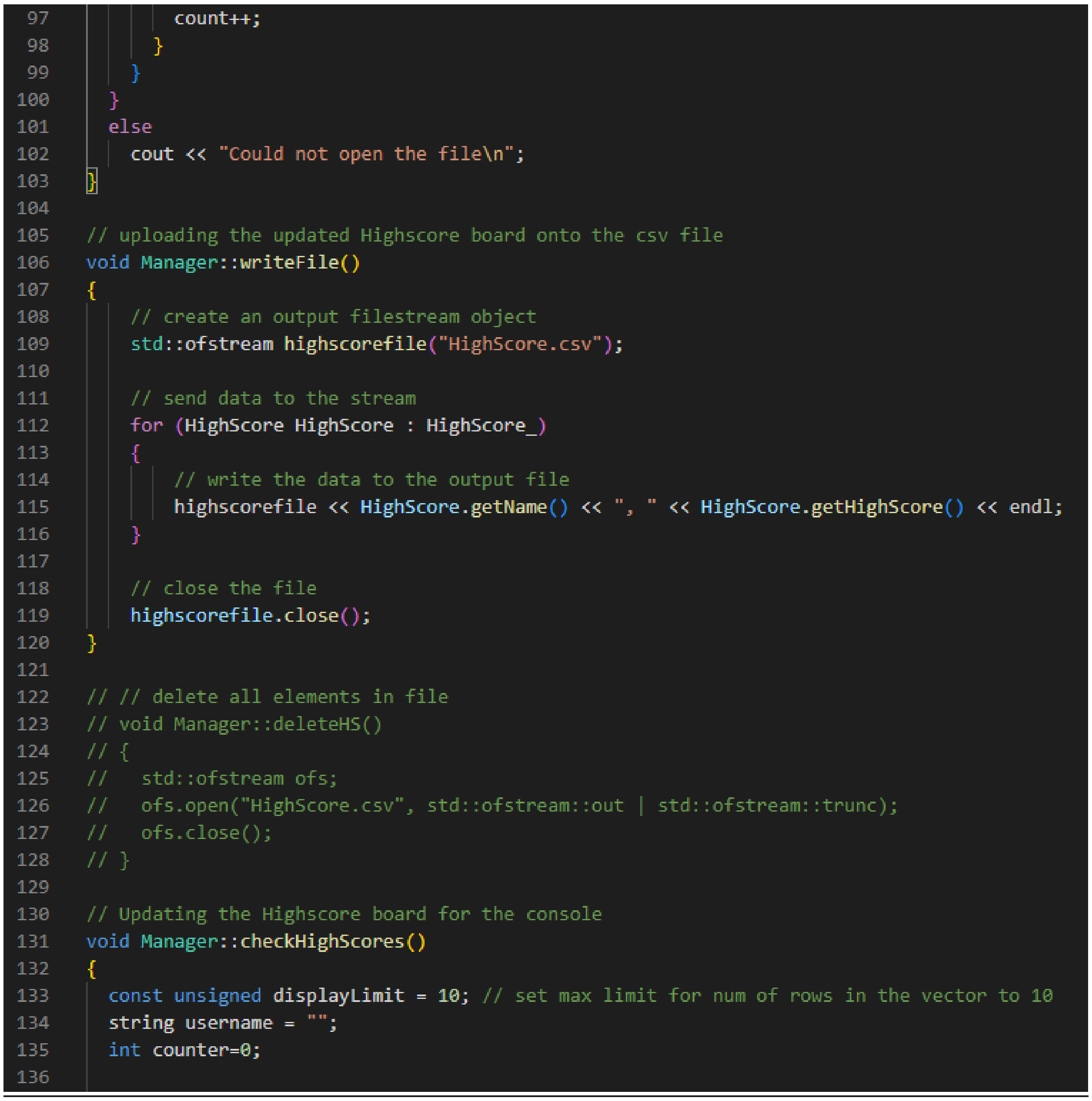
Manager.cpp

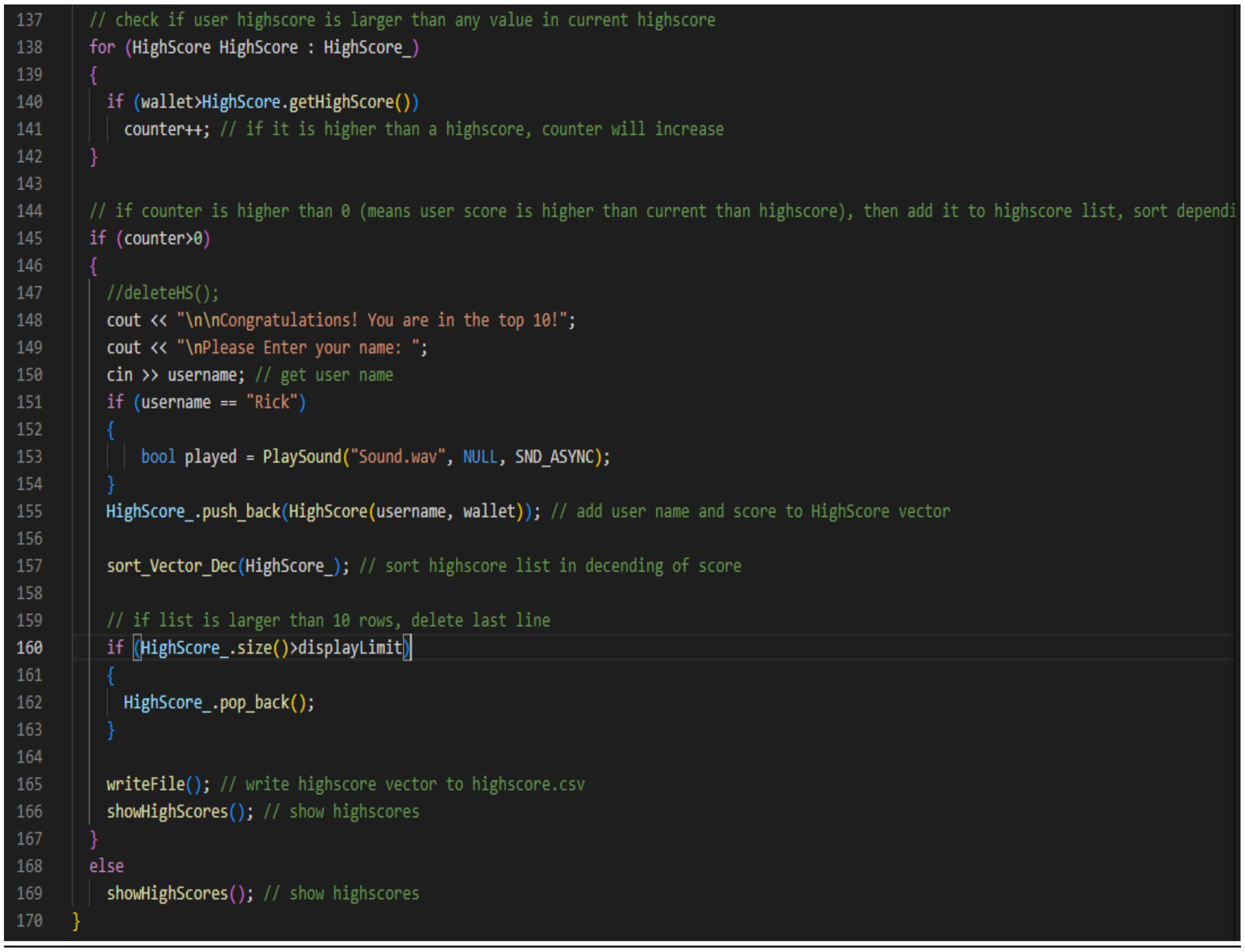


```cpp
#include <cstdlib>
#include <cctype>
#include <fstream>
#include <iomanip>
#include <iostream>
#include <locale>
#include <sstream>
#include <string>
#include <vector>
#include <random>
#include <algorithm>
#include <stdlib.h>
#include <Windows.h>

#include "Team.h"
#include "Manager.h"
#include "Match.h"
#include "Betting.h"

using namespace std;

int getRandomNumber(int , int); // initialise to get random numbers

//  Default constructor:
Manager::Manager() {}

//  Destructor:
Manager::~Manager() {}

// retrieve the World Cup csv file into the program
void Manager::loadFromFile()
{
  string line, word;

  fstream file("WorldCup.csv", ios::in);
  if (file.is_open())
  {
    while (getline(file, line))
    {
      stringstream str(line);

      std::string countryName = "";
      int countryScore = 0;
      size_t count = 0; // keep track of token count

      while (getline(str, word, ','))
      {
        if (count == 0) // 0 Means its reading country now
          countryName = word;
```

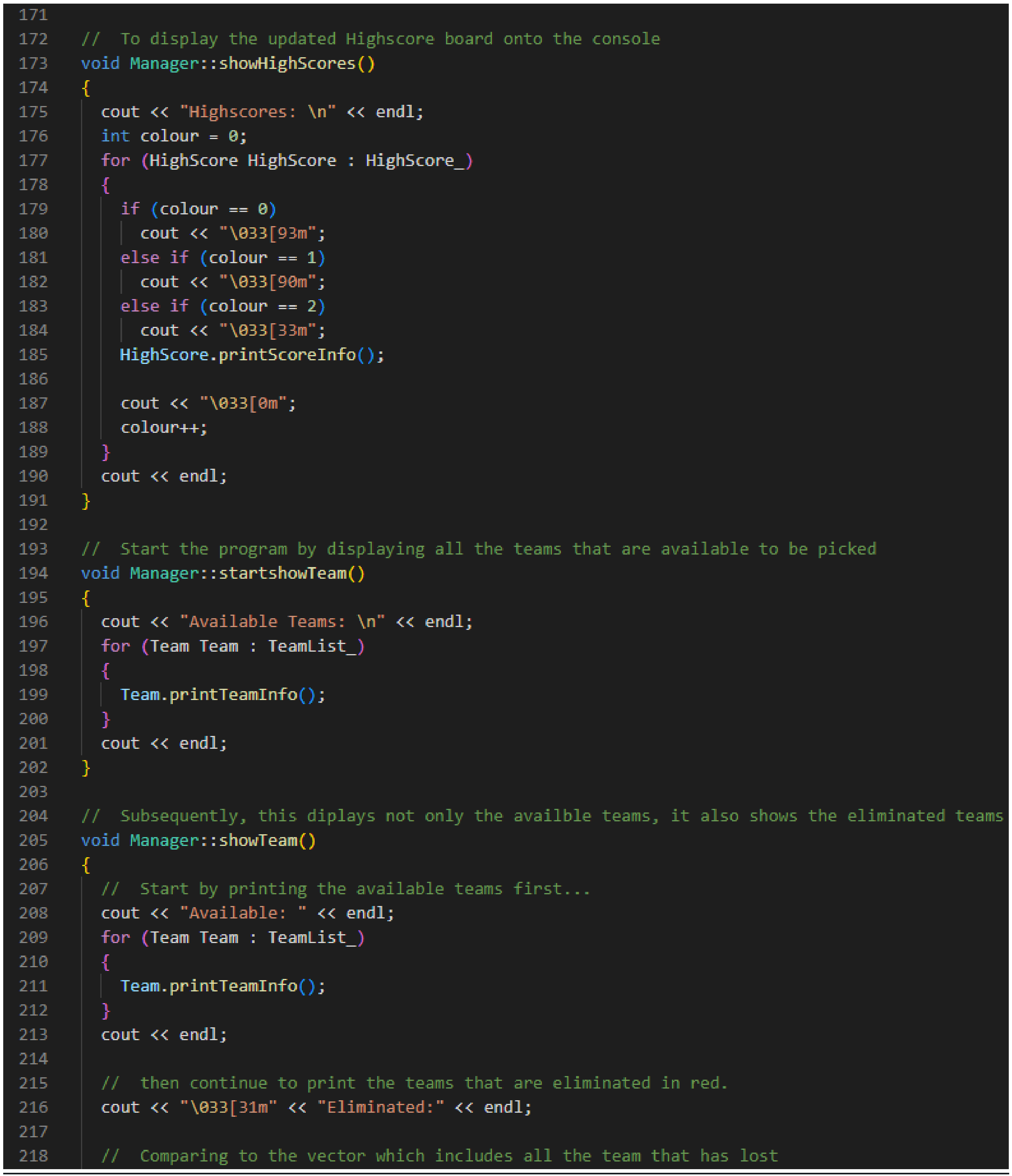


```
            else if (count == 1) // 1 means its reading the score now
            {
              // read score = end of line = u got all the required info
              countryScore = std::stoi(word); // convert string to int

              // create Team here and push into container
              TeamList_.push_back(Team(countryName, countryScore));
              count = 0;
            }
            count++;
          }
        }
      }
      else
        cout << "Could not open the file\n";
    }

    // retrieve the current Highscore csv file
    void Manager::loadHighScore()
    {
      string line, word;

      fstream file("HighScore.csv", ios::in);
      if (file.is_open())
      {
        while (getline(file, line))
        {
          stringstream str(line);

          std::string hsName = "";
          int hsScore = 0;
          size_t count = 0; // keep track of token count


          while (getline(str, word, ','))
          {
            if (count == 0) // 0 Means its reading country now
              hsName = word;
            else if (count == 1) // 1 means its reading the score now
            {
              // read score = end of line = u got all the required info
              hsScore = std::stoi(word); // convert string to int

              // create highscore and push into container
              HighScore_.push_back(HighScore(hsName, hsScore));
              count = 0;
            }
```

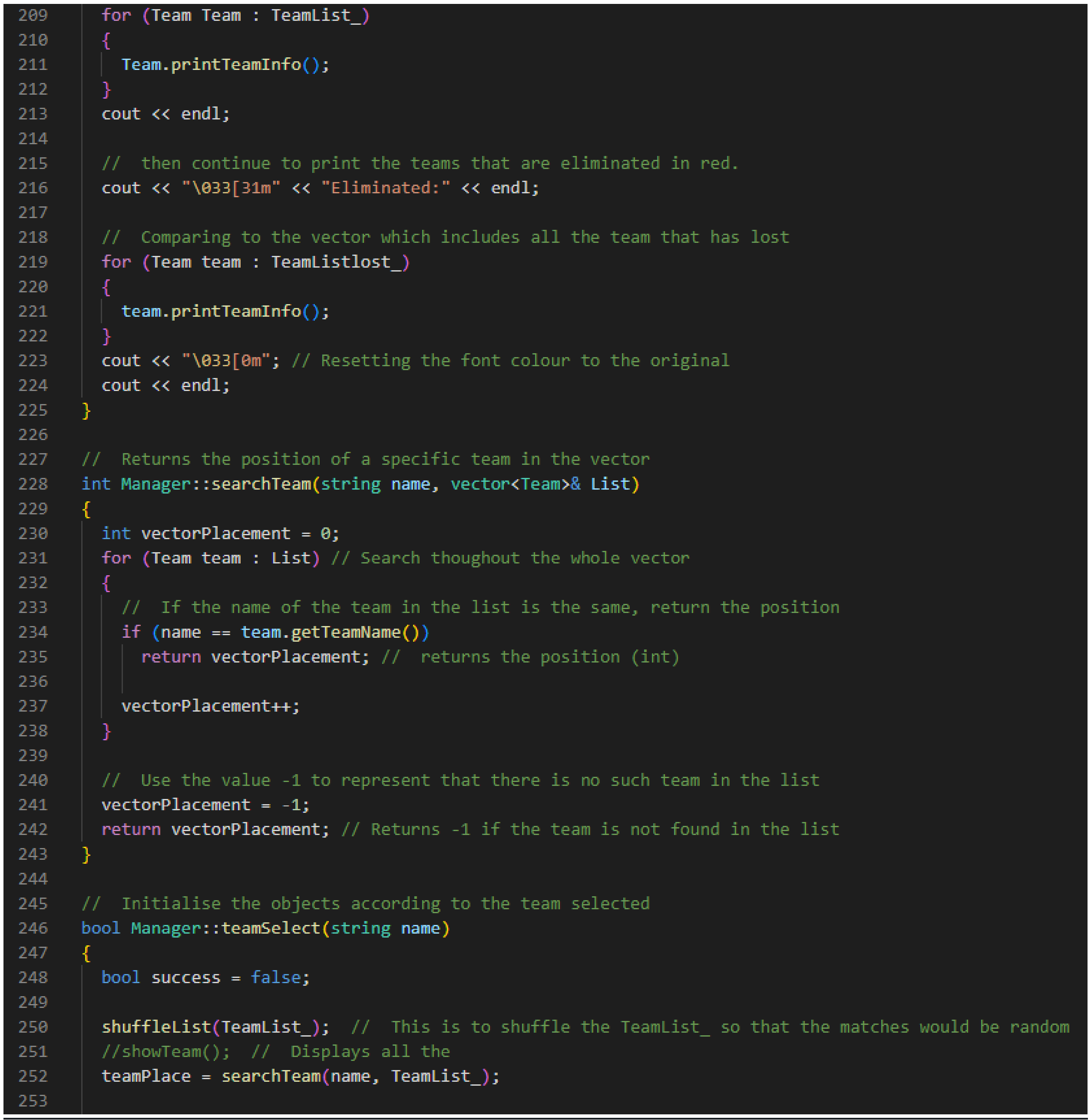

```cpp
                count++;
            }
        }
    }
    else
        cout << "Could not open the file\n";
}

// uploading the updated Highscore board onto the csv file
void Manager::writeFile()
{
    // create an output filestream object
    std::ofstream highscorefile("HighScore.csv");

    // send data to the stream
    for (HighScore HighScore : HighScore_)
    {
        // write the data to the output file
        highscorefile << HighScore.getName() << ", " << HighScore.getHighScore() << endl;
    }

    // close the file
    highscorefile.close();
}

// // delete all elements in file
// void Manager::deleteHS()
// {
//   std::ofstream ofs;
//   ofs.open("HighScore.csv", std::ofstream::out | std::ofstream::trunc);
//   ofs.close();
// }

// Updating the Highscore board for the console
void Manager::checkHighScores()
{
  const unsigned displayLimit = 10; // set max limit for num of rows in the vector to 10
  string username = "";
  int counter=0;

```



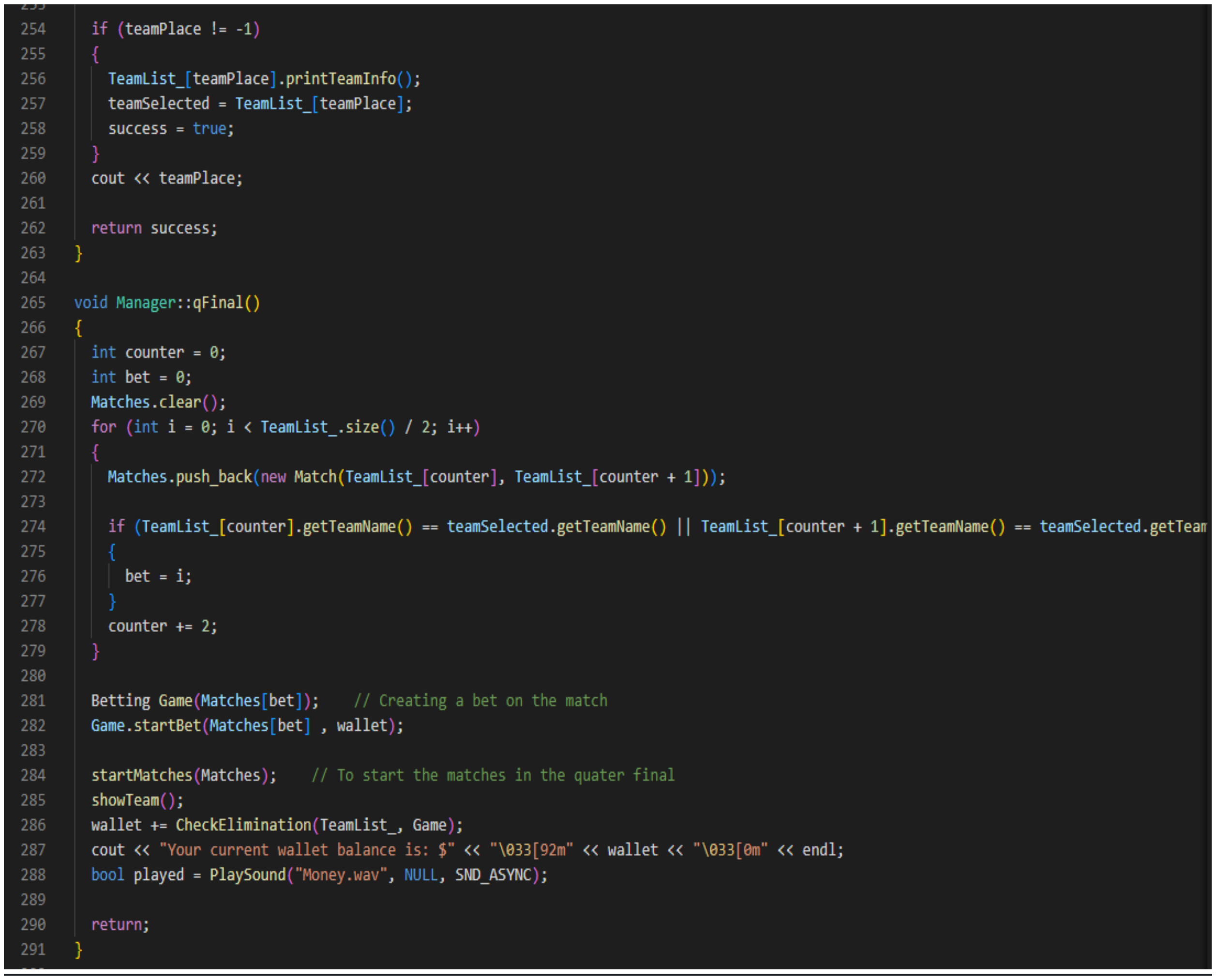
```
    // check if user highscore is larger than any value in current highscore
    for (HighScore HighScore : HighScore_)
    {
        if (wallet>HighScore.getHighScore())
            counter++; // if it is higher than a highscore, counter will increase
    }

    // if counter is higher than 0 (means user score is higher than current than highscore), then add it to highscore list, sort dependi
    if (counter>0)
    {
        //deleteHS();
        cout << "\n\nCongratulations! You are in the top 10!";
        cout << "\nPlease Enter your name: ";
        cin >> username; // get user name
        if (username == "Rick")
        {
                bool played = PlaySound("Sound.wav", NULL, SND_ASYNC);
        }
        HighScore_.push_back(HighScore(username, wallet)); // add user name and score to HighScore vector

        sort_Vector_Dec(HighScore_); // sort highscore list in decending of score

        // if list is larger than 10 rows, delete last line
        if (HighScore_.size()>displayLimit)
        {
            HighScore_.pop_back();
        }

        writeFile(); // write highscore vector to highscore.csv
        showHighScores(); // show highscores
    }
    else
        showHighScores(); // show highscores
}
```



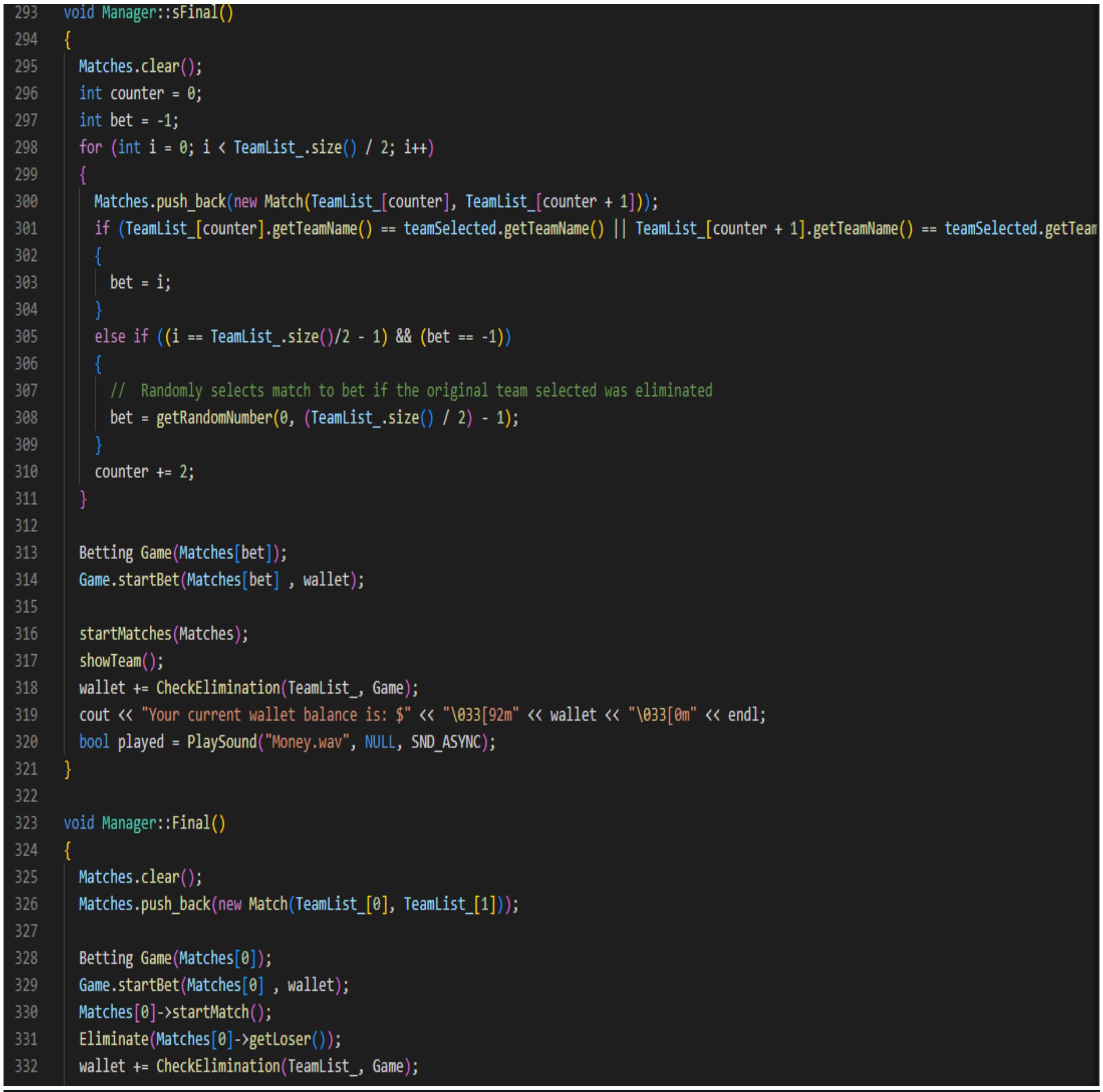
```cpp

//  To display the updated Highscore board onto the console
void Manager::showHighScores()
{
  cout << "Highscores: \n" << endl;
  int colour = 0;
  for (HighScore HighScore : HighScore_)
  {
    if (colour == 0)
      cout << "\033[93m";
    else if (colour == 1)
      cout << "\033[90m";
    else if (colour == 2)
      cout << "\033[33m";
    HighScore.printScoreInfo();

    cout << "\033[0m";
    colour++;
  }
  cout << endl;
}

//  Start the program by displaying all the teams that are available to be picked
void Manager::startshowTeam()
{
  cout << "Available Teams: \n" << endl;
  for (Team Team : TeamList_)
  {
    Team.printTeamInfo();
  }
  cout << endl;
}

//  Subsequently, this diplays not only the availble teams, it also shows the eliminated teams
void Manager::showTeam()
{
  //  Start by printing the available teams first...
  cout << "Available: " << endl;
  for (Team Team : TeamList_)
  {
    Team.printTeamInfo();
  }
  cout << endl;

  //  then continue to print the teams that are eliminated in red.
  cout << "\033[31m" << "Eliminated:" << endl;

  //  Comparing to the vector which includes all the team that has lost
```



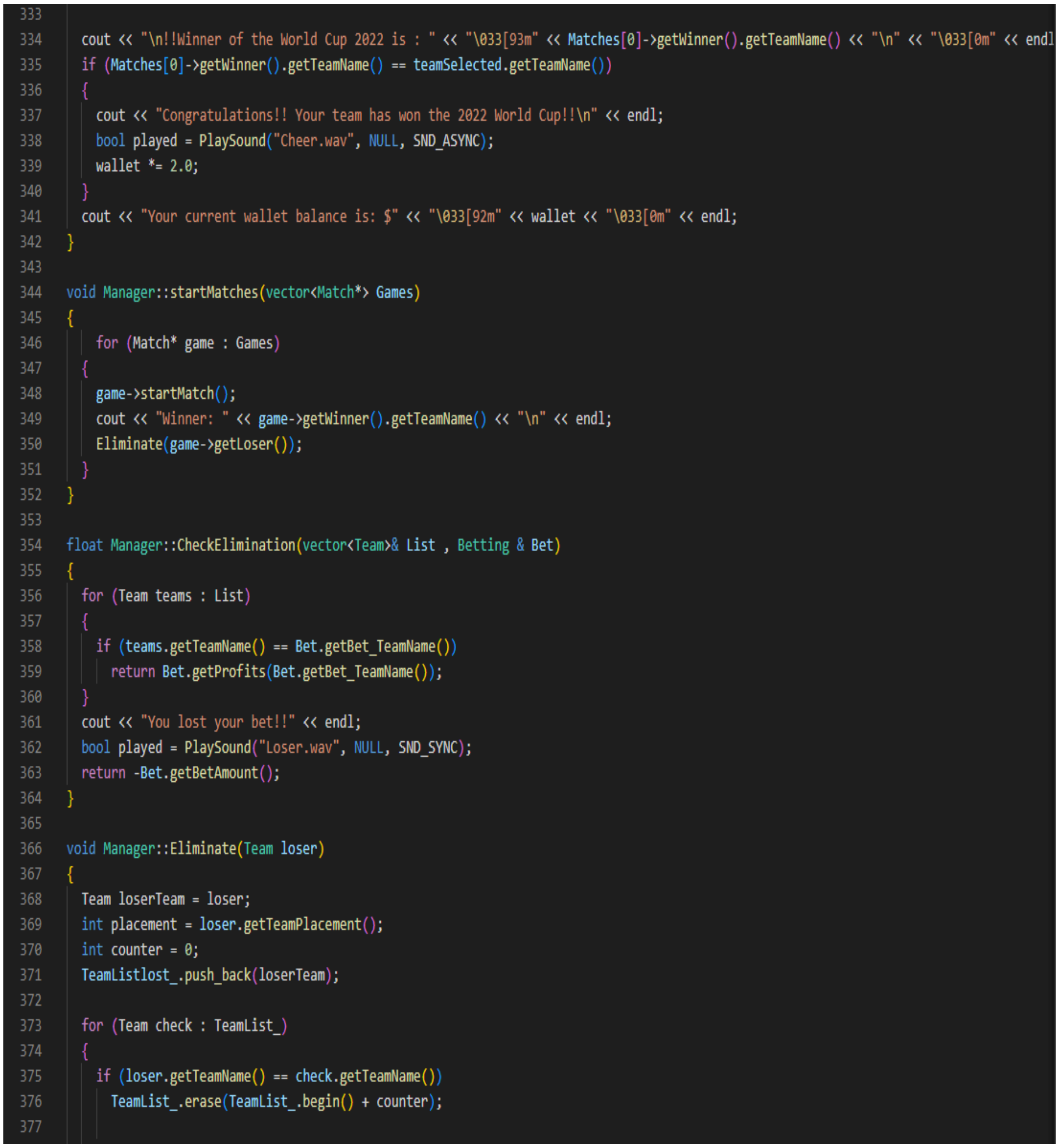
```cpp
  for (Team Team : TeamList_)
  {
    Team.printTeamInfo();
  }
  cout << endl;

  //  then continue to print the teams that are eliminated in red.
  cout << "\033[31m" << "Eliminated:" << endl;

  //  Comparing to the vector which includes all the team that has lost
  for (Team team : TeamListlost_)
  {
    team.printTeamInfo();
  }
  cout << "\033[0m"; // Resetting the font colour to the original
  cout << endl;
}

//  Returns the position of a specific team in the vector
int Manager::searchTeam(string name, vector<Team>& List)
{
  int vectorPlacement = 0;
  for (Team team : List) // Search thoughout the whole vector
  {
    //  If the name of the team in the list is the same, return the position
    if (name == team.getTeamName())
      return vectorPlacement; //  returns the position (int)

    vectorPlacement++;
  }

  //  Use the value -1 to represent that there is no such team in the list
  vectorPlacement = -1;
  return vectorPlacement; // Returns -1 if the team is not found in the list
}

//  Initialise the objects according to the team selected
bool Manager::teamSelect(string name)
{
  bool success = false;

  shuffleList(TeamList_);  //  This is to shuffle the TeamList_ so that the matches would be random
  //showTeam();  //  Displays all the
  teamPlace = searchTeam(name, TeamList_);

```



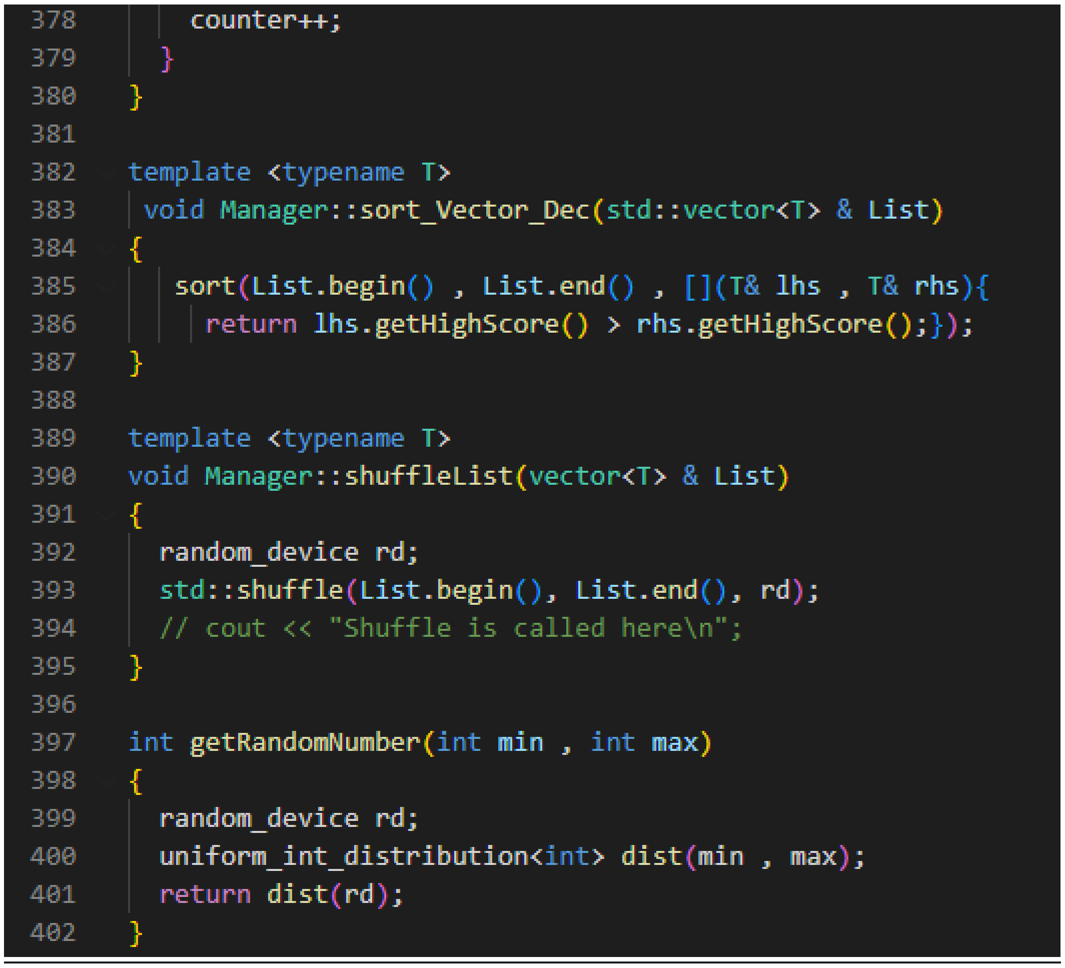
```cpp
    if (teamPlace != -1)
    {
      TeamList_[teamPlace].printTeamInfo();
      teamSelected = TeamList_[teamPlace];
      success = true;
    }
    cout << teamPlace;

    return success;
  }

  void Manager::qFinal()
  {
    int counter = 0;
    int bet = 0;
    Matches.clear();
    for (int i = 0; i < TeamList_.size() / 2; i++)
    {
      Matches.push_back(new Match(TeamList_[counter], TeamList_[counter + 1]));

      if (TeamList_[counter].getTeamName() == teamSelected.getTeamName() || TeamList_[counter + 1].getTeamName() == teamSelected.getTeam
      {
        bet = i;
      }
      counter += 2;
    }

    Betting Game(Matches[bet]);    // Creating a bet on the match
    Game.startBet(Matches[bet] , wallet);

    startMatches(Matches);    // To start the matches in the quater final
    showTeam();
    wallet += CheckElimination(TeamList_, Game);
    cout << "Your current wallet balance is: $" << "\033[92m" << wallet << "\033[0m" << endl;
    bool played = PlaySound("Money.wav", NULL, SND_ASYNC);

    return;
  }
```



```cpp
void Manager::sFinal()
{
  Matches.clear();
  int counter = 0;
  int bet = -1;
  for (int i = 0; i < TeamList_.size() / 2; i++)
  {
    Matches.push_back(new Match(TeamList_[counter], TeamList_[counter + 1]));
    if (TeamList_[counter].getTeamName() == teamSelected.getTeamName() || TeamList_[counter + 1].getTeamName() == teamSelected.getTeam
    {
      bet = i;
    }
    else if ((i == TeamList_.size()/2 - 1) && (bet == -1))
    {
      //  Randomly selects match to bet if the original team selected was eliminated
      bet = getRandomNumber(0, (TeamList_.size() / 2) - 1);
    }
    counter += 2;
  }

  Betting Game(Matches[bet]);
  Game.startBet(Matches[bet] , wallet);

  startMatches(Matches);
  showTeam();
  wallet += CheckElimination(TeamList_, Game);
  cout << "Your current wallet balance is: $" << "\033[92m" << wallet << "\033[0m" << endl;
  bool played = PlaySound("Money.wav", NULL, SND_ASYNC);
}

void Manager::Final()
{
  Matches.clear();
  Matches.push_back(new Match(TeamList_[0], TeamList_[1]));

  Betting Game(Matches[0]);
  Game.startBet(Matches[0] , wallet);
  Matches[0]->startMatch();
  Eliminate(Matches[0]->getLoser());
  wallet += CheckElimination(TeamList_, Game);
```

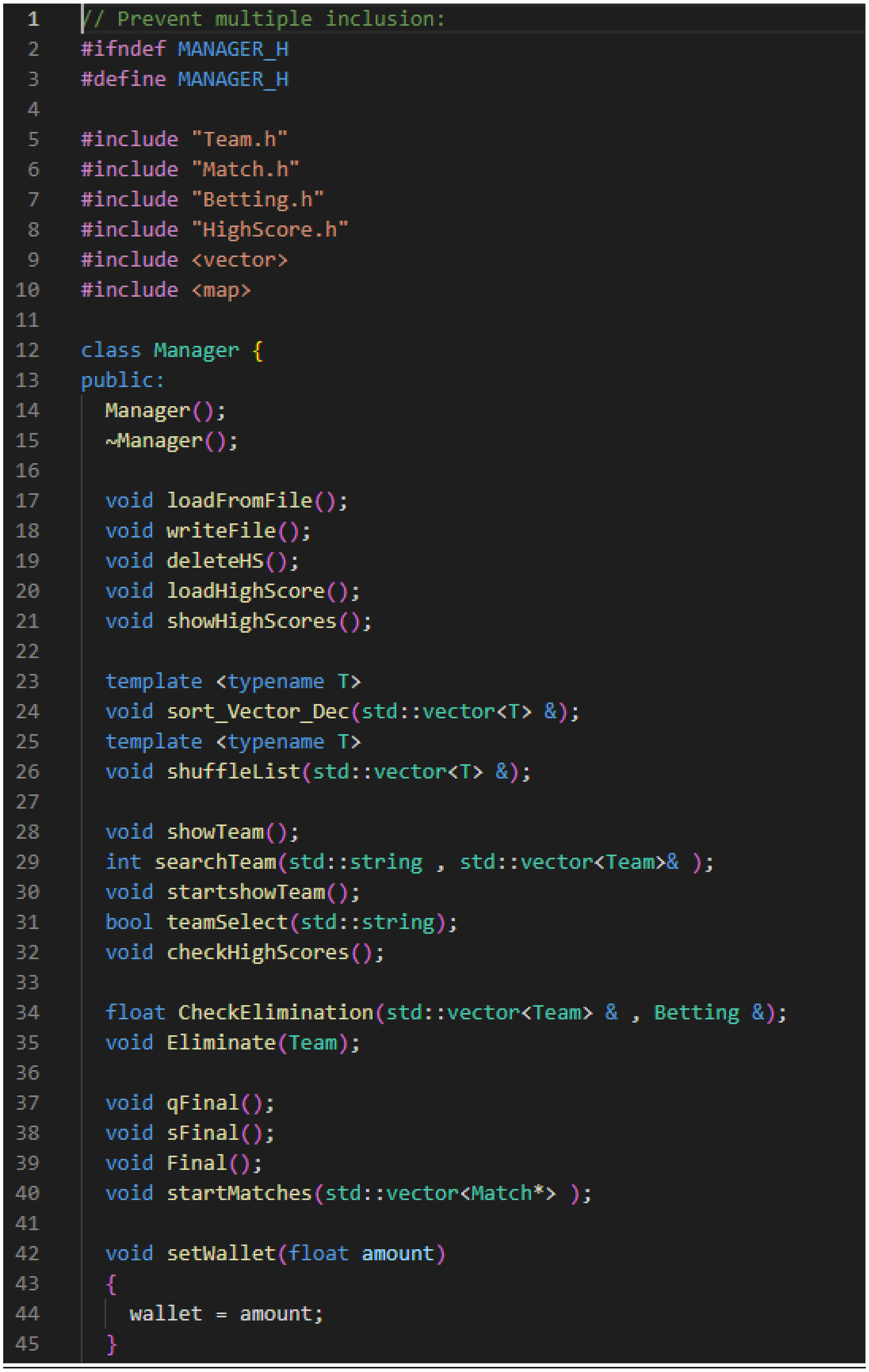


```cpp

    cout << "\n!!Winner of the World Cup 2022 is : " << "\033[93m" << Matches[0]->getWinner().getTeamName() << "\n" << "\033[0m" << endl
    if (Matches[0]->getWinner().getTeamName() == teamSelected.getTeamName())
    {
        cout << "Congratulations!! Your team has won the 2022 World Cup!!\n" << endl;
        bool played = PlaySound("Cheer.wav", NULL, SND_ASYNC);
        wallet *= 2.0;
    }
    cout << "Your current wallet balance is: $" << "\033[92m" << wallet << "\033[0m" << endl;
}

void Manager::startMatches(vector<Match*> Games)
{
        for (Match* game : Games)
    {
        game->startMatch();
        cout << "Winner: " << game->getWinner().getTeamName() << "\n" << endl;
        Eliminate(game->getLoser());
    }
}

float Manager::CheckElimination(vector<Team>& List , Betting & Bet)
{
    for (Team teams : List)
    {
        if (teams.getTeamName() == Bet.getBet_TeamName())
            return Bet.getProfits(Bet.getBet_TeamName());
    }
    cout << "You lost your bet!!" << endl;
    bool played = PlaySound("Loser.wav", NULL, SND_SYNC);
    return -Bet.getBetAmount();
}

void Manager::Eliminate(Team loser)
{
    Team loserTeam = loser;
    int placement = loser.getTeamPlacement();
    int counter = 0;
    TeamListlost_.push_back(loserTeam);

    for (Team check : TeamList_)
    {
        if (loser.getTeamName() == check.getTeamName())
            TeamList_.erase(TeamList_.begin() + counter);

```

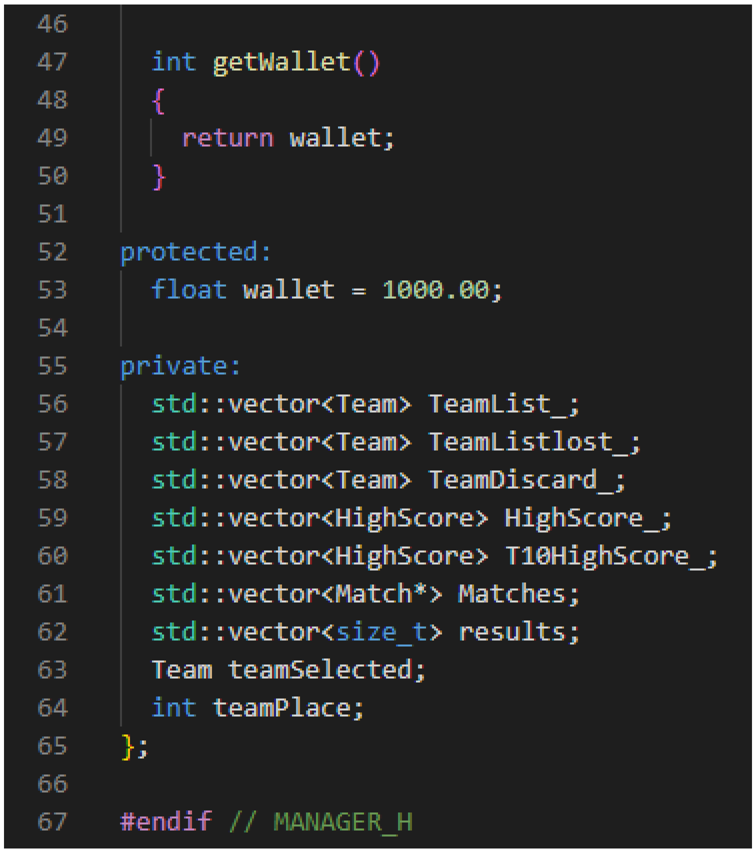


```cpp
        counter++;
    }
}

template <typename T>
 void Manager::sort_Vector_Dec(std::vector<T> & List)
{
    sort(List.begin() , List.end() , [](T& lhs , T& rhs){
        return lhs.getHighScore() > rhs.getHighScore();});
}

template <typename T>
void Manager::shuffleList(vector<T> & List)
{
    random_device rd;
    std::shuffle(List.begin(), List.end(), rd);
    // cout << "Shuffle is called here\n";
}

int getRandomNumber(int min , int max)
{
    random_device rd;
    uniform_int_distribution<int> dist(min , max);
    return dist(rd);
}
```

Manager.h

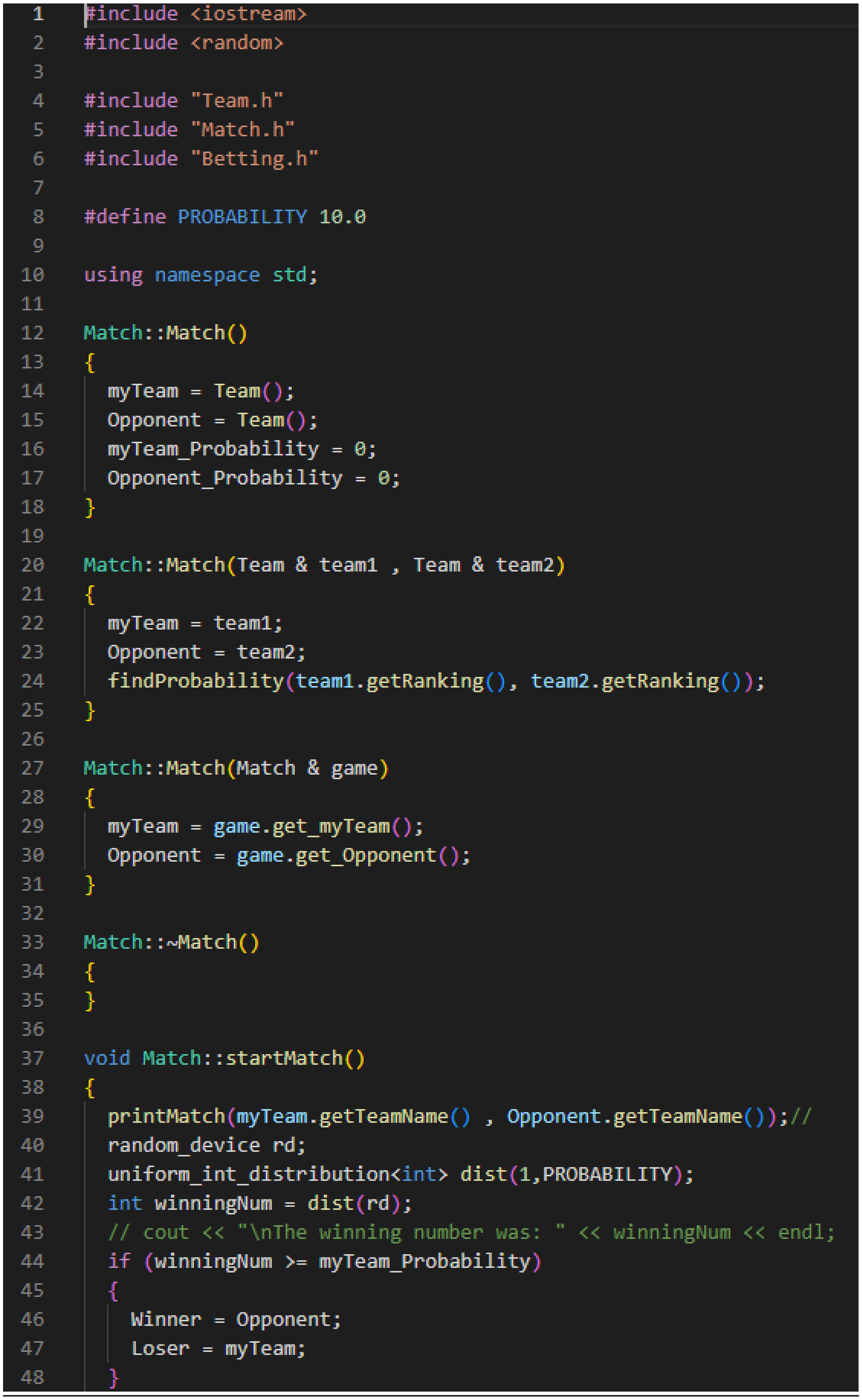


```cpp
// Prevent multiple inclusion:
#ifndef MANAGER_H
#define MANAGER_H

#include "Team.h"
#include "Match.h"
#include "Betting.h"
#include "HighScore.h"
#include <vector>
#include <map>

class Manager {
public:
  Manager();
  ~Manager();

  void loadFromFile();
  void writeFile();
  void deleteHS();
  void loadHighScore();
  void showHighScores();

  template <typename T>
  void sort_Vector_Dec(std::vector<T> &);
  template <typename T>
  void shuffleList(std::vector<T> &);

  void showTeam();
  int searchTeam(std::string , std::vector<Team>& );
  void startshowTeam();
  bool teamSelect(std::string);
  void checkHighScores();

  float CheckElimination(std::vector<Team> & , Betting &);
  void Eliminate(Team);

  void qFinal();
  void sFinal();
  void Final();
  void startMatches(std::vector<Match*> );

  void setWallet(float amount)
  {
    wallet = amount;
  }
```

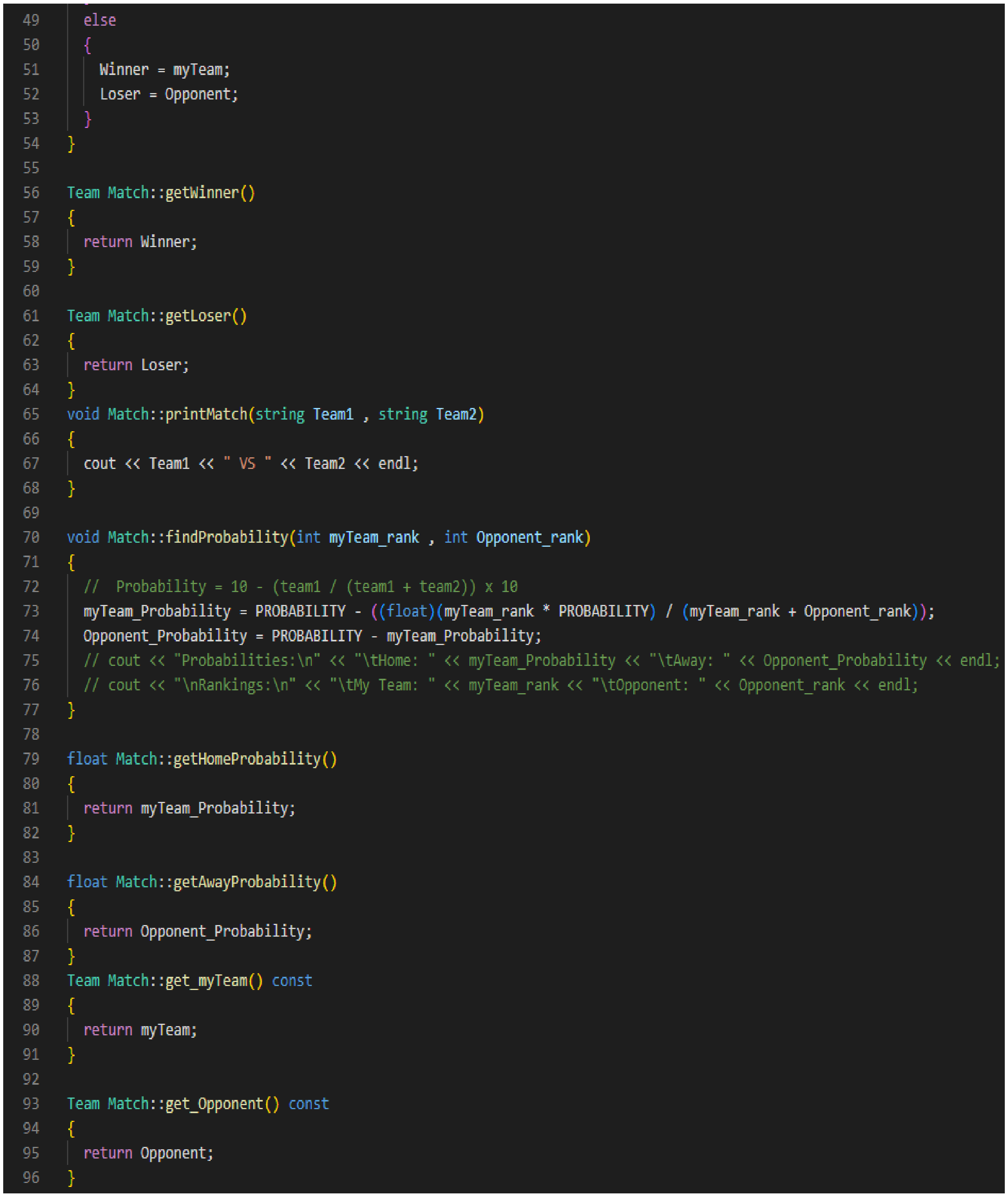


```cpp

    int getWallet()
    {
        return wallet;
    }

protected:
    float wallet = 1000.00;

private:
    std::vector<Team> TeamList_;
    std::vector<Team> TeamListlost_;
    std::vector<Team> TeamDiscard_;
    std::vector<HighScore> HighScore_;
    std::vector<HighScore> T10HighScore_;
    std::vector<Match*> Matches;
    std::vector<size_t> results;
    Team teamSelected;
    int teamPlace;
};

#endif // MANAGER_H
```

Match.cpp

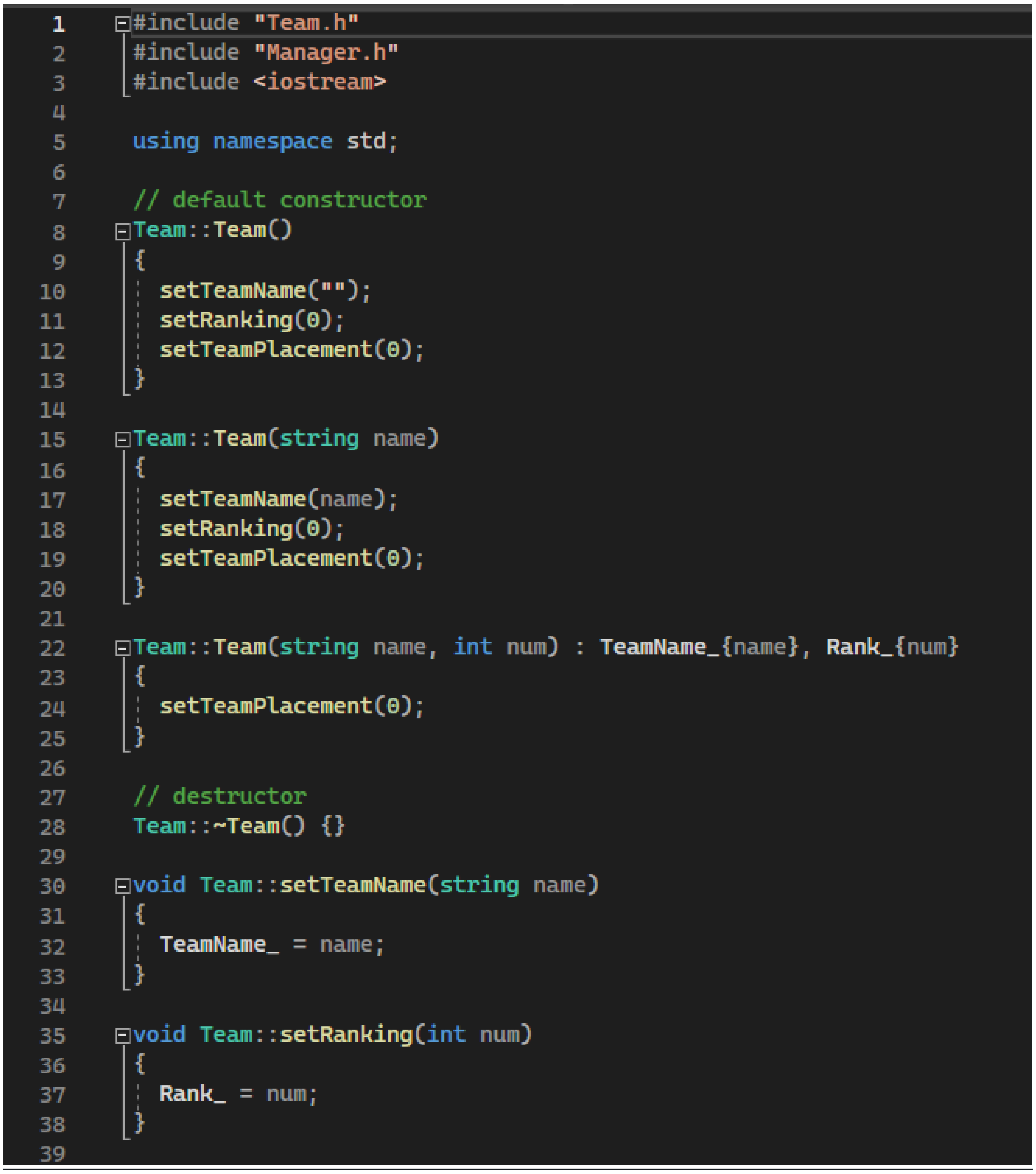


```cpp
#include <iostream>
#include <random>

#include "Team.h"
#include "Match.h"
#include "Betting.h"

#define PROBABILITY 10.0

using namespace std;

Match::Match()
{
  myTeam = Team();
  Opponent = Team();
  myTeam_Probability = 0;
  Opponent_Probability = 0;
}

Match::Match(Team & team1 , Team & team2)
{
  myTeam = team1;
  Opponent = team2;
  findProbability(team1.getRanking(), team2.getRanking());
}

Match::Match(Match & game)
{
  myTeam = game.get_myTeam();
  Opponent = game.get_Opponent();
}

Match::~Match()
{
}

void Match::startMatch()
{
  printMatch(myTeam.getTeamName() , Opponent.getTeamName());//
  random_device rd;
  uniform_int_distribution<int> dist(1,PROBABILITY);
  int winningNum = dist(rd);
  // cout << "\nThe winning number was: " << winningNum << endl;
  if (winningNum >= myTeam_Probability)
  {
    Winner = Opponent;
    Loser = myTeam;
  }
```

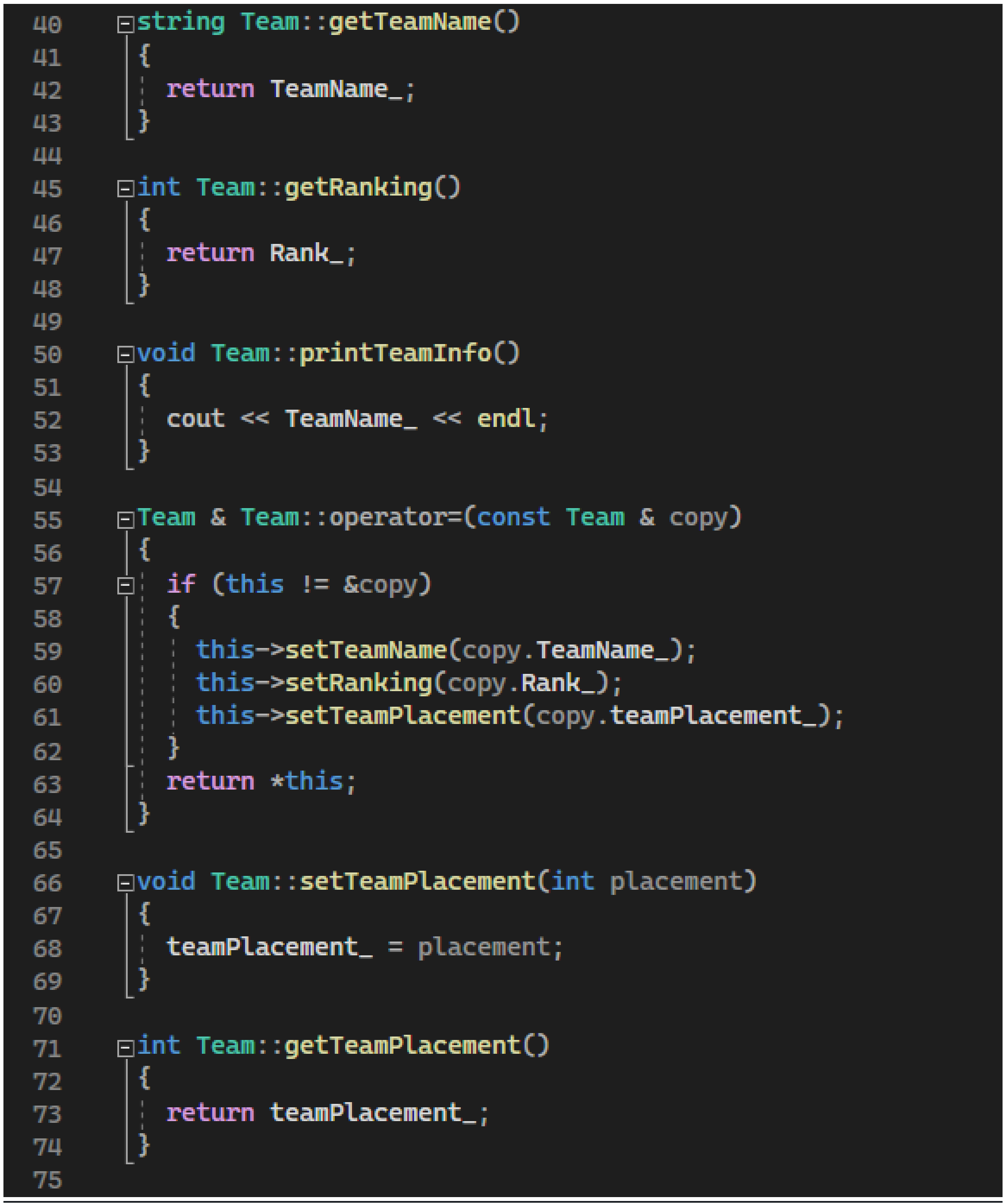


```cpp
  else
  {
    Winner = myTeam;
    Loser = Opponent;
  }
}

Team Match::getWinner()
{
  return Winner;
}

Team Match::getLoser()
{
  return Loser;
}
void Match::printMatch(string Team1 , string Team2)
{
  cout << Team1 << " VS " << Team2 << endl;
}

void Match::findProbability(int myTeam_rank , int Opponent_rank)
{
  //  Probability = 10 - (team1 / (team1 + team2)) x 10
  myTeam_Probability = PROBABILITY - ((float)(myTeam_rank * PROBABILITY) / (myTeam_rank + Opponent_rank));
  Opponent_Probability = PROBABILITY - myTeam_Probability;
  // cout << "Probabilities:\n" << "\tHome: " << myTeam_Probability << "\tAway: " << Opponent_Probability << endl;
  // cout << "\nRankings:\n" << "\tMy Team: " << myTeam_rank << "\tOpponent: " << Opponent_rank << endl;
}

float Match::getHomeProbability()
{
  return myTeam_Probability;
}

float Match::getAwayProbability()
{
  return Opponent_Probability;
}
Team Match::get_myTeam() const
{
  return myTeam;
}

Team Match::get_Opponent() const
{
  return Opponent;
}
```

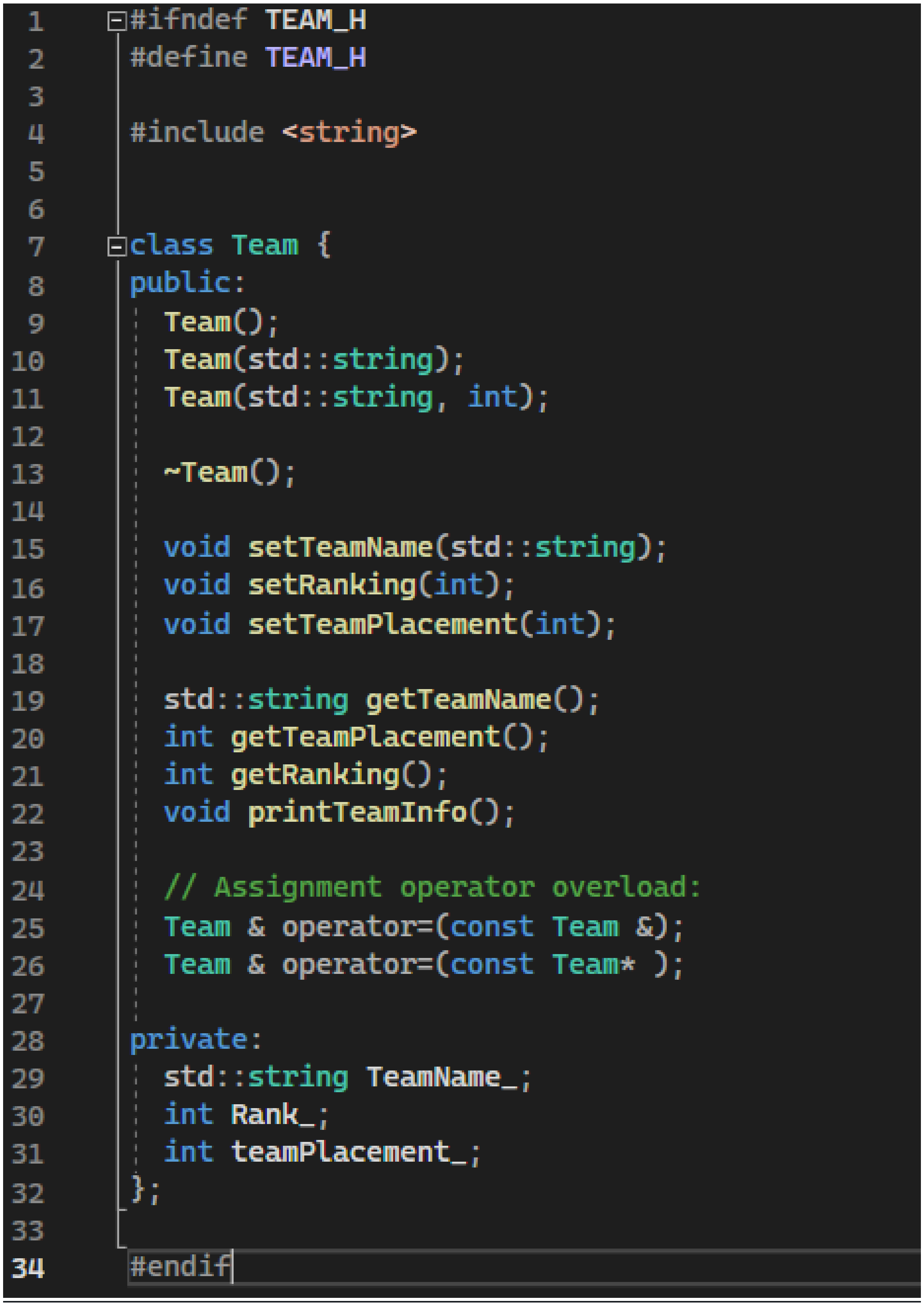
Team.cpp



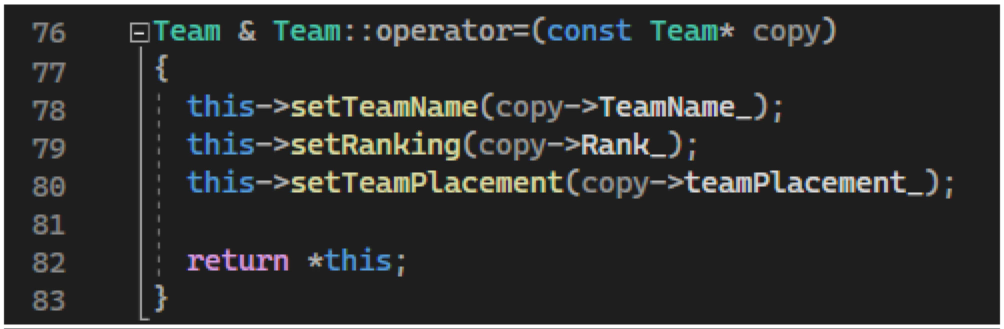
```cpp
#include "Team.h"
#include "Manager.h"
#include <iostream>

using namespace std;

// default constructor
Team::Team()
{
    setTeamName("");
    setRanking(0);
    setTeamPlacement(0);
}

Team::Team(string name)
{
    setTeamName(name);
    setRanking(0);
    setTeamPlacement(0);
}

Team::Team(string name, int num) : TeamName_{name}, Rank_{num}
{
    setTeamPlacement(0);
}

// destructor
Team::~Team() {}

void Team::setTeamName(string name)
{
    TeamName_ = name;
}

void Team::setRanking(int num)
{
    Rank_ = num;
}

```


```cpp
string Team::getTeamName()
{
	return TeamName_;
}

int Team::getRanking()
{
	return Rank_;
}

void Team::printTeamInfo()
{
	cout << TeamName_ << endl;
}

Team & Team::operator=(const Team & copy)
{
	if (this != &copy)
	{
		this->setTeamName(copy.TeamName_);
		this->setRanking(copy.Rank_);
		this->setTeamPlacement(copy.teamPlacement_);
	}
	return *this;
}

void Team::setTeamPlacement(int placement)
{
	teamPlacement_ = placement;
}

int Team::getTeamPlacement()
{
	return teamPlacement_;
}

```

Team.h

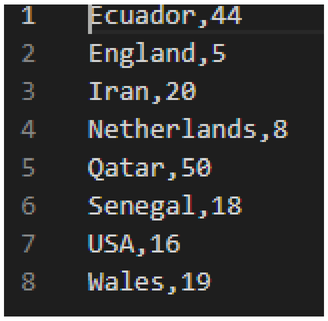


```cpp
#ifndef TEAM_H
#define TEAM_H

#include <string>


class Team {
public:
  Team();
  Team(std::string);
  Team(std::string, int);

  ~Team();

  void setTeamName(std::string);
  void setRanking(int);
  void setTeamPlacement(int);

  std::string getTeamName();
  int getTeamPlacement();
  int getRanking();
  void printTeamInfo();

  // Assignment operator overload:
  Team & operator=(const Team &);
  Team & operator=(const Team* );

private:
  std::string TeamName_;
  int Rank_;
  int teamPlacement_;
};

#endif
```



```cpp
Team & Team::operator=(const Team* copy)
{
    this->setTeamName(copy->TeamName_);
    this->setRanking(copy->Rank_);
    this->setTeamPlacement(copy->teamPlacement_);

    return *this;
}
```

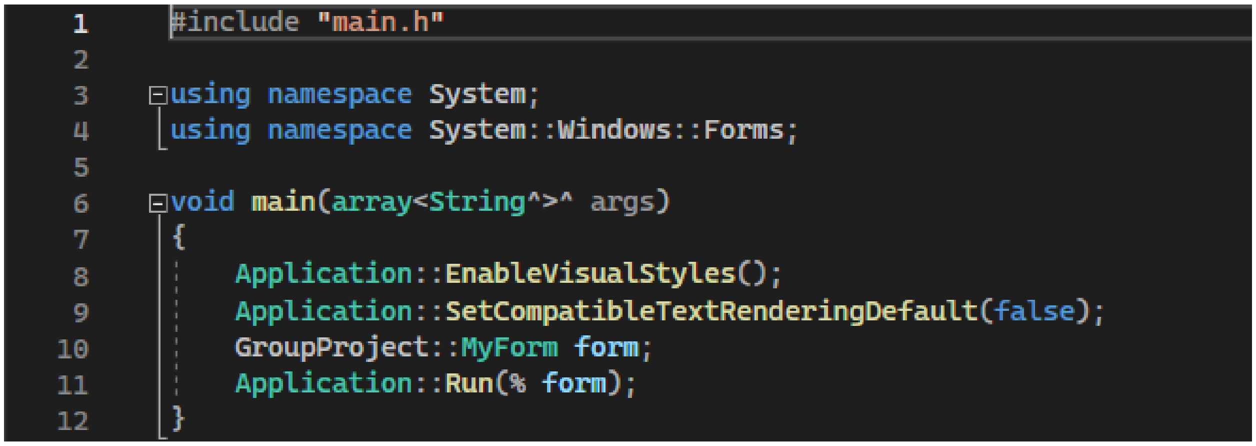
WorldCup.csv



```
Ecuador,44
England,5
Iran,20
Netherlands,8
Qatar,50
Senegal,18
USA,16
Wales,19
```

GUI

Main.cpp



```cpp
#include "main.h"

using namespace System;
using namespace System::Windows::Forms;

void main(array<String^>^ args)
{
    Application::EnableVisualStyles();
    Application::SetCompatibleTextRenderingDefault(false);
    GroupProject::MyForm form;
    Application::Run(% form);
}
```