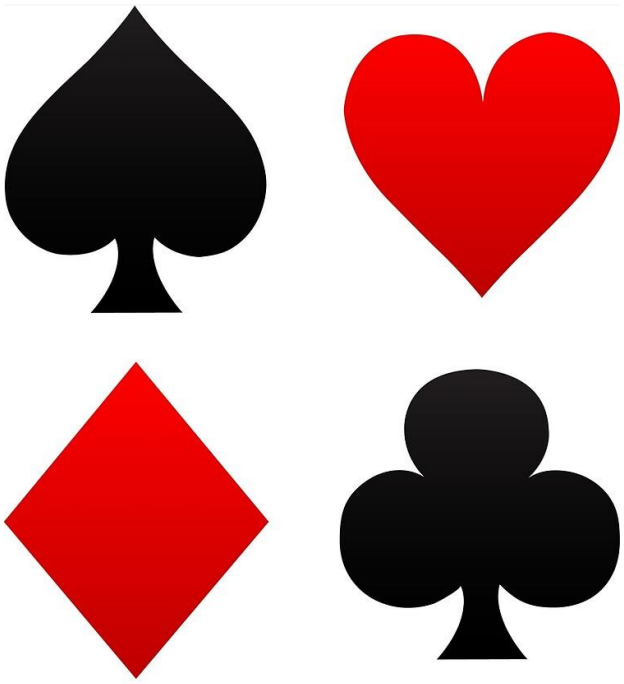

CIS 2220

SPRING SEMESTER 2020

PROFESSOR BRADLEY SWARD

MIKE APREZA

MARCH 7th, 2020

TABLE OF CONTENTS

Other File(s)

Text File(s)

PlayerList.txt

```
randomdude99 1000 5 16 9
yellow_hare 1001 0 2 5
tech_guy21 1002 10 26 3
c0013r00 1003 15 0 1
BlueBox22 1004 0 3 0
RedSerpent04 1005 20 47 2
pink_unicorn 1006 1 6 7
RealBeast45 1007 60 122 1
RedRobinYum 1008 24 9 0
```

Complete C++ Code (with comment statements), including Header, Implementation, and Driver files(s).

Driver File

main.cpp

```
/*
	Name:	Mike Apreza
	File:	main.cpp
	Date:	02/10/2020
		Updated: 03/07/2020
	Class:	CIS 2220
*/

#include "Player.h"
#include "PlayerManager.h"
#include "BlackjackGame.cpp"
#include "HighLowGame.cpp"
#include <iostream>
#include <string>

int main()
{
	// Create PlayerManager to load file
	PlayerManager* instance = PlayerManager::getInstance();
	// Create int variables
	int choice, id, index = 0;
	double cred, bet, outcome;
	// Create string variable
	std::string user;
	// bool variable to see if user has selected a player
	bool haveCurrentPlayer = false;

	// Menu
	do
	{
		system("CLS");
		std::cout << "\tWelcome Manager.\n\n"
			<< " 1. Add a Player\n"
			<< " 2. Remove a Player\n"
			<< " 3. Modify a Players\n"
			<< " 4. List Players\n"
			<< " 5. Select a New Current Player\n"
			<< " 6. Show Current Player\n"
			<< " 7. Play Blackjack Hand\n"
			<< " 8. Play High - Low Hand\n"
			<< " 9. Exit\n"
			<< "Enter your choice: ";
		std::cin >> choice;

		switch (choice)
		{
			case 1:
			{
				system("CLS");
				std::cout << "Enter the username of the new player: ";
				std::cin >> user;
				std::cout << "Enter the credits of the new player: ";
				std::cin >> cred;
				instance->addUser(user, cred);
				break;
```

```
                        }
                        case 2:
                        {
                                system("CLS");
                                std::cout << "Enter the ID number of the player you wish to
remove: ";
                                std::cin >> id;
                                instance->removeUser(id);
                                break;
                        }
                        case 3:
                        {
                                system("CLS");
                                std::cout << "Enter the ID number of the player you wish to
modify: ";
                                std::cin >> id;
                                instance->modifyUser(id);
                                break;
                        }
                        case 4:
                        {
                                system("CLS");
                                instance->listUsers();
                                break;
                        }
                        case 5:
                        {
                                system("CLS");
                                // Select new current player
                                index = instance->selectNewPlayer();
                                haveCurrentPlayer = true;
                                break;
                        }
                        case 6:
                        {
                                system("CLS");
                                if (haveCurrentPlayer == false)
                                {
                                        std::cout << "Please select a player." << std::endl;
                                        break;
                                }
                                // Show current player
                                instance->showPlayer();
                                break;
                        }
                        case 7:
                        {
                                system("CLS");
                                // Check to see if a player has been picked
                                if (haveCurrentPlayer == false)
                                {
                                        std::cout << "Please select a player." << std::endl;
                                        system("PAUSE");
                                        break;
                                }
                                // Check to see if there are any credits
                                if (instance->returnCPCred() == 0)
                                {
                                        std::cout << "You do not have enough credits to play.
Please add more credits.\n";
                                        system("PAUSE");
```

```
                        }
                        else
                        {
                                std::cout << "The current player has " << instance-
>returnCPCred() << " credits.\n";
                                do
                                {
                                        std::cout << "Enter a valid bet: ";
                                        std::cin >> bet;
                                } while ((bet <= 0) || (bet > instance->returnCPCred()));
                                // Set appropriate number with function
                                if (bet == instance->returnCPCred())
                                {
                                        outcome = BlackjackGame(bet);
                                        if (outcome == -999)
                                        {
                                                instance->modifyRecord(index, 0, bet);
                                        }
                                        else
                                        {
                                                instance->modifyRecord(outcome, index, 0);
                                        }
                                }
                                else
                                {
                                        outcome = BlackjackGame(bet);
                                        if (outcome == -999)
                                        {
                                                instance->modifyRecord(index, 1, bet);
                                        }
                                        else
                                        {
                                                instance->modifyRecord(outcome, index, 1);
                                        }
                                }
                        }
                        break;
                }
                case 8:
                {
                        system("CLS");
                        // Check to see if a player has been picked
                        if (haveCurrentPlayer == false)
                        {
                                std::cout << "Please select a player." << std::endl;
                                system("PAUSE");
                                break;
                        }
                        // Check to see if there are any credits
                        if (instance->returnCPCred() == 0)
                        {
                                std::cout << "You do not have enough credits to play.
Please add more credits.\n";
                                system("PAUSE");
                        }
                        else
                        {
                                std::cout << "The current player has " << instance-
>returnCPCred() << " credits.\n";
                                do
                                {
```

```
                                        std::cout << "Enter a valid bet: ";
                                        std::cin >> bet;
                                } while ((bet <= 0) || (bet > instance->returnCPCred()));
                                // Set appropriate number with function
                                if (bet == instance->returnCPCred())
                                        instance->modifyRecord(HighLowGame(bet), index,
0);
                                else
                                        instance->modifyRecord(HighLowGame(bet), index,
1);
                                }
                                break;
                        }
                        case 9:
                                break;
                        default:
                        {
                                std::cout << "Enter a valid number." << std::endl;
                                break;
                        }
                }
                system("PAUSE");
        } while (choice != 9);
        return 0;
}
```

Header Files

PlayerManager.h

```
/*
    Name:      Mike Apreza
    File:      PlayerManager.h
    Date:      02/10/2020
        Updated: 03/06/2020
    Class:     CIS 2220
*/

#include <vector>
#include "Player.h"
#pragma once

// Singleton Design
class PlayerManager
{
    // private member variables and methods
    private:
        std::vector<Player*> players;
        Player currentP;
        void modifyFile();
        // pointer to PlayerManager object
        static PlayerManager* Instance;
        // constructor
        PlayerManager();
    // public member functions
    public:
        // returns pointer to PlayerManager object
        static PlayerManager* getInstance();
        // member functions
        void addUser(const std::string &user, double creds);
        void removeUser(int id);
        void listUsers();
        void modifyUser(int id);
        void showPlayer();
        int selectNewPlayer();
        void modifyRecord(double outcome, int index, int specificUpdate);
        void modifyRecord(int index, int specificUpdate, double bet);
        double returnCPCred();
};
```

Player.h

```
/*
	Name:	Mike Apreza
	File:	Player.h
	Date:	02/10/2020
		Updated: 03/02/2020
	Class:	CIS 2220
*/

#include <string>
#pragma once

class Player
{
	// private member variables
	private:
		std::string username;
		int IDnumber;
		double credits;
		int wins;
		int losses;
	// public member functions
	public:
		// constructors
		Player();
		Player(const std::string &user, int id, double creds, int win, int loss);
		// setters
		void setUsername(const std::string &user);
		void setID(int id);
		void setCredits(double cred);
		void fixCredits(double cred);
		void setWins(int w);
		void setLosses(int l);
		// getters
		std::string getUsername();
		int getID();
		double getCredits();
		int getWins();
		int getLosses();
};
```

CardGame.h

```
/*
	Name:	Mike Apreza
	File:	CardGame.h
	Date:	03/07/2020
	Class:	CIS 2220
*/

#pragma once
#include <string>

class CardGame
{
	// private
	private:
		// hold player's bet
		double bet = 0.0;
	// public
	public:
		// sets and returns player's bet
		void setBet(double b);
		double getBet();
		// return random number between 1 and 13
		int getCard();
		// returns what string to print out (for cards)
		std::string realCard(int card);
		// compares cards and returns a number (children will redefine and appropriate
action will be carried out)
		virtual int compare(int card1, int card2);
};
```

Blackjack.h

```
/*
	Name:	Mike Apreza
	File:	Blackjack.h
	Date:	03/06/2020
	Class:	CIS 2220
*/

#pragma once
#include "CardGame.h"

class Blackjack : public CardGame
{
	private:
		int dealerCard1 = 0;
		int dealerCard2 = 0;
		int dealerHand = 0;
		int playerCard1 = 0;
		int playerCard2 = 0;
		int playerHand = 0;
	public:
		// constructor
		Blackjack();
		int getDealerCard1();
		int getDealerCard2();
		int getPlayerCard1();
		void fixPlayerHand();
		int getPlayerCard2();
		int getDealerHand();
		void addToDealerHand(int value);
		int getPlayerHand();
		void addToPlayerHand(int value);
		virtual int compare(int playerH, int dealerH) override;
};
```

HighLow.h

```
/*
	Name:	Mike Apreza
	File:	HighLow.h
	Date:	03/07/2020
	Class:	CIS 2220
*/

#pragma once
#include "CardGame.h"

class HighLow : public CardGame
{
	// private member variables
	private:
		int currentCard = 0;
		int nextCard = 0;
		int pay = 0;
		double pays[13][2] =
		{{1.0, 1.0}, {10.7, 1.1}, {5.3, 1.1},
		{3.5, 1.1}, {2.6, 1.3}, {2.1, 1.5},
		{1.87, 1.87}, {1.5, 2.1}, {1.3, 2.6},
		{1.1, 3.5}, {1.1, 5.3}, {1.1, 10.7}, {1.0, 1.0}};
	public:
		// constructor
		HighLow(double bet);
		// member functions
		void setCurrentCard(int card);
		int getCurrentCard();
		void setNextCard(int card);
		int getNextCard();
		void setPay(double multiplier);
		int getPay();
		// get appropriate multiplier
		double getMultiplier(char type);
		virtual int compare(int current, int next) override;
};
```

Implementation Files

PlayerManager.cpp

```
/*
	Name:	Mike Apreza
	File:	PlayerManager.cpp
	Date:	02/10/2020
		Updated: 03/02/2020
	Class:	CIS 2220
*/

#include "PlayerManager.h"
#include "Player.h"
#include <iostream>
#include <fstream>
#include <vector>
#include <string>
#include <cstdlib>
#include <iomanip>

PlayerManager* PlayerManager::Instance(nullptr);

PlayerManager::PlayerManager()
{
	// create variables
	std::string user;
	int id;
	double creds;
	int win, loss;
	// open file and read
	std::ifstream playersFile;
	playersFile.open("PlayerList.txt");
	if (playersFile.fail())
	{
		std::cout << "File Opening Error. Please Try Again." << std::endl;
		exit(1);
	}
	else
	{
		while (playersFile >> user >> id >> creds >> win >> loss)
			players.push_back(new Player(user, id, creds, win, loss));
		playersFile.close();
	}
}

// return pointer to PlayerManager
PlayerManager* PlayerManager::getInstance()
{
	if (Instance == nullptr)
	{
		Instance = new PlayerManager();
	}
	return Instance;
}

void PlayerManager::addUser(const std::string &user, double creds)
{
	// generate ID
	int idNum;
	bool unique;
```

```
	do
	{
		unique = true;
		idNum = (rand() % 9000) + 1000;
		for (unsigned int i = 0; i < players.size(); ++i)
			if (players[i]->getID() == idNum)
				unique = false;
	} while (unique == false);
	// Add new player to "players" vector
	players.push_back(new Player(user, idNum, creds, 0, 0));
	// Open file and add user
	std::ofstream file;
	file.open("PlayerList.txt", std::ios_base::app);
	if (file.fail())
	{
		std::cout << "File Opening Error. Please Try Again." << std::endl;
		exit(1);
	}
	else
		file << user << " " << idNum << " " << creds << " " << 0 << " " << 0 <<
std::endl;
	file.close();
	std::cout << "User Successfully Added.\n";
}

void PlayerManager::listUsers()
{
	std::cout << std::left << std::setw(20) << "Username" << "ID Number\tCredits\t  Wins\t
Losses" << std::endl;
	for (unsigned int m = 0; m < players.size(); ++m)
	{
		std::cout << std::left << std::setw(20) << players[m]->getUsername()
			<< players[m]->getID() << "\t"
			<< players[m]->getCredits() << "\t  "
			<< players[m]->getWins() << "\t "
			<< players[m]->getLosses() <<  std::endl;
	}
}

void PlayerManager::removeUser(int id)
{
	int index = -1;
	for (unsigned int k = 0; k < players.size(); ++k)
	{
		if (players[k]->getID() == id)
			index = k;
	}
	if (index == -1)
		std::cout << "A user with the ID: " << id << " was not found. Please try again."
<< std::endl;
	else
	{
		players.erase(players.begin() + index);
		// Modify file
		modifyFile();
		std::cout << "Player successfully removed.\n";
	}
}

void PlayerManager::modifyUser(int id)
{
```

```
	int index = -1, choice;
	double creds;
	std::string user;
	for (unsigned int k = 0; k < players.size(); ++k)
	{
		if (players[k]->getID() == id)
			index = k;
	}
	if (index == -1)
		std::cout << "A user with the ID: " << id << " was not found. Please try again."
<< std::endl;
	else
	{
		std::cout << "Would you like to modify the username or credits available?\n"
			<< " 1. Username\n"
			<< " 2. Credits\n"
			<< "Enter your choice: ";
		std::cin >> choice;
		while ((choice != 1) && (choice != 2))
		{
			std::cout << "Enter a valid number: ";
			std::cin >> choice;
		}
		switch (choice)
		{
			case 1:
			{
				std::cout << "Enter the new username of the player: ";
				std::cin >> user;
				players[index]->setUsername(user);
				break;
			}
			case 2:
			{
				std::cout << "Enter the amount of credits of the player: ";
				std::cin >> creds;
				players[index]->setCredits(creds);
				break;
			}
		}

		// Modify file
		modifyFile();
		std::cout << "Player successfully modified.\n";
	}
}

void PlayerManager::showPlayer()
{
	std::cout << "Current Player:\n" << " Username: " << currentP.getUsername() <<
std::endl;
	std::cout << " ID Number: " << currentP.getID() << std::endl;
	std::cout << " Credits: " << currentP.getCredits() << std::endl;
	std::cout << " Wins: " << currentP.getWins() << std::endl;
	std::cout << " Losses: " << currentP.getLosses() << std::endl;
}

int PlayerManager::selectNewPlayer()
{
	static int choice;
```

```
	std::cout << "Which player would you like to select? (Enter corresponding number): " <<
std::endl;
	std::cout << "   Username\t\tID" << std::endl;
	for (unsigned int k = 0; k < players.size(); ++k)
		std::cout << k + 1 << ". " << players[k]->getUsername() << "\t\t" << players[k]-
>getID() << std::endl;
	do
	{
		std::cin >> choice;
	} while ((choice < 1) && (choice > players.size()));
	--choice;
	currentP.setUsername(players[choice]->getUsername());
	currentP.setID(players[choice]->getID());
	currentP.setCredits(players[choice]->getCredits());
	currentP.setWins(players[choice]->getWins());
	currentP.setLosses(players[choice]->getLosses());
	return choice;
}

void PlayerManager::modifyRecord(double outcome, int index, int specificUpdate)
{
	if (outcome == 0)
	{
		// 0 is a draw so nothing occurs (not even the credits are modified)
	}
	else if (outcome > 0)
	{
		// Update wins in currentP object
		currentP.setWins(currentP.getWins() + 1);
		// Update wins in players vector
		players[index]->setWins(currentP.getWins());
		// If specificUpdate == 1 then it means that the bet was NOT all of the
available credits
		if (specificUpdate == 1)
		{
			// Update credits in currentP object
			currentP.fixCredits(outcome);
			// Update credits in players vector
			players[index]->fixCredits(outcome);
		}
		else
		{
			// Update credits in currentP object
			currentP.setCredits(outcome);
			// Update credits in players vector
			players[index]->setCredits(outcome);
		}
		// Modify file
		modifyFile();
	}
	else
	{
		// Update losses in currentP object
		currentP.setLosses(currentP.getLosses() + 1);
		// Update losses in players vector
		players[index]->setLosses(currentP.getLosses());
		// If specificUpdate == 1 then it means that the bet was NOT all of the
available credits
		if (specificUpdate == 1)
		{
			// Update credits in currentP object
```

```
                        currentP.fixCredits(outcome);
                        // Update credits in players vector
                        players[index]->fixCredits(outcome);
                }
                else
                {
                        // Update credits in currentP object
                        currentP.setCredits(0);
                        // Update credits in players vector
                        players[index]->setCredits(0);
                }
                // Modify file
                modifyFile();
        }
}

void PlayerManager::modifyRecord(int index, int specificUpdate, double bet)
{
        // If user did NOT bet all of their credits
        if (specificUpdate == 1)
        {
                static double newCredits;
                newCredits = -(bet / 2);
                currentP.fixCredits(newCredits);
                players[index]->fixCredits(newCredits);
        }
        else
        {
                currentP.setCredits(bet / 2);
                players[index]->setCredits(currentP.getCredits());
        }
        // Modfy file
        modifyFile();
}

double PlayerManager::returnCPCred()
{ return currentP.getCredits(); }

void PlayerManager::modifyFile()
{
        std::ofstream file;
        file.open("PlayerList.txt");
        if (file.fail())
        {
                std::cout << "File Opening Error. Please Try Again." << std::endl;
                exit(1);
        }
        else
        {
                Player* ptr;
                for (unsigned int t = 0; t < players.size(); ++t)
                {
                        ptr = players.at(t);
                        file << ptr->getUsername() << " " << ptr->getID() << " " << ptr-
>getCredits()
                                << " " << ptr->getWins() << " " << ptr->getLosses() << std::endl;
                }
                file.close();
                std::cout << "Player successfully modified.\n";
        }
}
```

Player.cpp

```
/*
	Name:	Mike Apreza
	File:	Player.cpp
	Date:	02/10/2020
		Updated: 03/02/2020
	Class:	CIS 2220
*/

#include "Player.h"
#include <iostream>
#include <string>

// constructors
Player::Player() :username(" "), IDnumber(0), credits(0), wins(0), losses(0)
{ /* Nothing */ }

Player::Player(const std::string &user, int id, double creds, int win, int loss) :
	username(user), IDnumber(id), credits(creds), wins(win), losses(loss)
{ /* Nothing */ }

// member functions
void Player::setUsername(const std::string &user)
{ username = user; }

void Player::setID(int id)
{ IDnumber = id; }

void Player::setCredits(double cred)
{ credits = cred; }

void Player::fixCredits(double cred)
{ credits += cred; }

void Player::setWins(int w)
{ wins = w; }

void Player::setLosses(int l)
{ losses = l; }

std::string Player::getUsername()
{ return username; }

int Player::getID()
{ return IDnumber; }

double Player::getCredits()
{ return credits; }

int Player::getWins()
{ return wins; }

int Player::getLosses()
{ return losses; }
```

CardGame.cpp

```
/*
    Name:      Mike Apreza
    File:      CardGame.cpp
    Date:      03/07/2020
    Class:     CIS 2220
*/

#include "CardGame.h"
#include <stdlib.h>

void CardGame::setBet(double b)
{ bet = b; }

double CardGame::getBet()
{ return bet; }

int CardGame::getCard()
{
        static int card;
        card = (rand() % 13) + 1;
        return card;
}

std::string CardGame::realCard(int card)
{
    switch (card)
    {
      case 1:
        return "A";
      case 11:
        return "J";
      case 12:
        return "Q";
      case 13:
        return "K";
      default:
          return std::to_string(card);
    }
}

int CardGame::compare(int card1, int card2)
{
    if (card1 > card2)
        return 1;
    else if (card1 < card2)
        return 0;
    else
        return -1;
}
```

Blackjack.cpp

```
/*
	Name:	Mike Apreza
	File:	Blackjack.cpp
	Date:	03/07/2020
	Class:	CIS 2220
*/

#include "Blackjack.h"
#include <iostream>

Blackjack::Blackjack()
{
	// Give dealer and player 2 cards each
	dealerCard1 = getCard();
	dealerCard2 = getCard();
	playerCard1 = getCard();
	playerCard2 = getCard();
	// sum hands
	if ((dealerCard1 + dealerCard2) == 2)
		dealerHand = 12;
	else if (dealerCard1 == 1)
		dealerHand += 11;
	else if (dealerCard2 == 1)
		dealerHand += 11;
	if (dealerCard1 > 10)
		dealerHand += 10;
	if (dealerCard2 > 10)
		dealerHand += 10;
	if ((dealerCard1 > 1) && (dealerCard1 < 11))
		dealerHand += dealerCard1;
	if ((dealerCard2 > 1) && (dealerCard2 < 11))
		dealerHand += dealerCard2;

	if (playerCard1 == 1)
		playerHand += 11;
	if (playerCard2 == 1)
		playerHand += 11;
	if (playerCard1 > 10)
		playerHand += 10;
	if (playerCard2 > 10)
		playerHand += 10;
	if ((playerCard1 > 1) && (playerCard1 < 11))
		playerHand += playerCard1;
	if ((playerCard2 > 1) && (playerCard2 < 11))
		playerHand += playerCard2;
}

int Blackjack::getDealerCard1()
{ return dealerCard1; }

int Blackjack::getDealerCard2()
{ return dealerCard2; }

int Blackjack::getPlayerCard1()
{ return playerCard1; }

void Blackjack::fixPlayerHand()
{ playerHand -= 10; }
```

```
int Blackjack::getPlayerCard2()
{ return playerCard2; }

int Blackjack::getDealerHand()
{ return dealerHand; }

void Blackjack::addToDealerHand(int value)
{ dealerHand += value; }

int Blackjack::getPlayerHand()
{ return playerHand; }

void Blackjack::addToPlayerHand(int value)
{ playerHand += value; }

int Blackjack::compare(int playerH, int dealerH)
{
	// Determine outcome
	if (dealerH > 21)
	{
		std::cout << "The dealers busts! You win!" << std::endl;
		system("PAUSE");
		return 1;
	}
	else if (dealerH == playerH)
	{
		std::cout << "You both have equal hands! It's a draw!" << std::endl;
		system("PAUSE");
		return 0;
	}
	else if (dealerH < playerH)
	{
		std::cout << "Your hand is closer to 21! You win!" << std::endl;
		system("PAUSE");
		return 2;
	}
	else if (dealerH > playerH)
	{
		std::cout << "The dealer's hand is closer to 21! You lose!" << std::endl;
		system("PAUSE");
		return -1;
	}
}
```

BlackjackGame.cpp

```
/*
    Name:      Mike Apreza
    File:      BlackjackGame.cpp
    Date:      03/07/2020
    Class:     CIS 2220
*/
#include "Blackjack.h"
#include <iostream>

static double BlackjackGame(double bet)
{
    int choice = 0, nextCard = 0;

    // Create Blackjack object
    Blackjack b;
    b.setBet(bet);

    // Show hands
    std::cout << "\t\tDealer: " << b.getDealerCard1() << "   ?\n\n"
        << "\tYour hand: " << b.realCard(b.getPlayerCard1()) << "   " <<
b.realCard(b.getPlayerCard2()) << std::endl;

    // Check if dealer has an ace or 10 for a chance at immediate blackjack, else continue
with the game
    if (((b.getDealerCard1() == 1) || (b.getDealerCard1() == 10)) && b.getDealerHand() == 21)
    {
        if (b.getPlayerHand() == 21)
        {
            std::cout << "\tThe dealer has a Blackjack! But so do you! It's a draw!" <<
std::endl;
            system("PAUSE");
            return 0;
        }
        else
        {
            std::cout << "\nThe dealer has a Blackjack! You do not! You lose!" << std::endl;
            system("PAUSE");
            return (-1 * b.getBet());
        }
    }

    // Ask player what value they want their ace to be if any are present
    if ((b.getPlayerCard1() == 1) && (b.getPlayerCard2() == 1))
    {
        std::cout << "\nYou have a Aces! Do you want the first to be worth 1 or 11? ";
        std::cin >> choice;
        while ((choice != 1) && (choice != 11))
        {
            std::cout << "Enter a valid choice: ";
            std::cin >> choice;
        }
        if (choice == 1)
            b.fixPlayerHand();

        std::cout << "What about the second (1 or 11)? ";
        std::cin >> choice;
        while ((choice != 1) && (choice != 11))
        {
            std::cout << "Enter a valid choice: ";
```

```
            std::cin >> choice;
        }
        if (choice == 1)
            b.fixPlayerHand();
        if (b.getPlayerHand() > 21)
        {
            std::cout << "Bust! You lose!" << std::endl;
            system("PAUSE");
            return (-1 * b.getBet());
        }
    }
    else if ((b.getPlayerCard1() == 1) || (b.getPlayerCard2() == 1))
    {
        std::cout << "\nYou have an Ace! Do you want it to be worth 1 or 11? ";
        std::cin >> choice;
        while ((choice != 1) && (choice != 11))
        {
            std::cout << "Enter a valid choice: ";
            std::cin >> choice;
        }
        if (choice == 1)
            b.fixPlayerHand();
    }

    // Ask player if they wish to surrender
    std::cout << "\nDo you wish to surrender (1/2 of your bet is lost) (1 for Yes, 2 for No)?
";
    std::cin >> choice;
    while ((choice != 1) && (choice != 2))
    {
        std::cout << "Enter a valid number: ";
        std::cin >> choice;
    }
    if (choice == 1)
    {
        b.setBet(b.getBet() / 2);
        std::cout << "You have lost half of your bet." << std::endl;
        system("PAUSE");
        return -999;
    }

    do
    {
        system("CLS");
        // Show hands
        std::cout << "\t\tDealer: " << b.getDealerCard1() << "   ?\n\n"
            << "\tYour hand total: " << b.getPlayerHand() << std::endl << std::endl;
        std::cout << "1. Hit (take another card)\n"
            << "2. Stand (take no more cards)\n"
            << "Enter your choice: ";
        std::cin >> choice;
        while ((choice != 1) && (choice != 2))
        {
            std::cout << "Enter a valid choice: ";
            std::cin >> choice;
        }
        if (choice == 1)
        {
            nextCard = b.getCard();
            std::cout << "You got a(n) ";
            if (nextCard == 1)
```

```
            {
                std::cout << "Ace. Do you wish it to be worth 1 or 11? ";
                std::cin >> choice;
                while ((choice != 1) && (choice != 11))
                {
                    std::cout << "Enter a valid number (1 or 11): ";
                    std::cin >> choice;
                }
                b.addToPlayerHand(choice);
            }
            else if ((nextCard == 11) || (nextCard == 12) || (nextCard == 13))
            {
                std::cout << b.realCard(nextCard) << ".\n";
                b.addToPlayerHand(10);
            }
            else
                std::cout << nextCard;
            b.addToPlayerHand(nextCard);
        }

        // Check if player busts
        if (b.getPlayerHand() > 21)
        {
            std::cout << "Bust! You lose!";
            system("PAUSE");
            return (-1 * b.getBet());
        }
    } while (choice != 2);

    // Turn over dealer's second card
    system("CLS");
    std::cout << "\t\tDealer: " << b.realCard(b.getDealerCard1()) << " " <<
b.realCard(b.getDealerCard2()) << std::endl;
    std::cout << "\n\n\tYour hand total: " << b.getPlayerHand() << std::endl << std::endl;

    if (b.getDealerHand() < 17)
    {
        nextCard = b.getCard();
        if ((b.getDealerHand() < 11) && (nextCard == 1))
            nextCard = 11;
        else if (nextCard > 10)
            nextCard = 10;
        b.addToDealerHand(nextCard);
        std::cout << "The dealer's hand is less than 17!"
            << "The dealer drew a card and now has the total of "<< b.getDealerHand() <<
std::endl;
    }

    // Determine outcome
    switch (b.compare(b.getPlayerHand(), b.getDealerHand()))
    {
        case 0:
            return 0;
        case 1:
            return b.getBet();
        case 2:
            return b.getBet();
        case -1:
            return (-1 * b.getBet());
    }
}
```

HighLowGame.cpp

```
/*
    Name:      Mike Apreza
    File:      HighLowGame.cpp
    Date:      03/07/2020
    Class:     CIS 2220
*/

#include <iostream>
#include "HighLow.h"

static double HighLowGame(double bet)
{
    int choice;
    char hORl = 'h';

    // create HighLow object
    HighLow HL(bet);

    // loop for game
    while (true)
    {
        system("CLS");
        // Display player's current card
        std::cout << "Your card: " << HL.realCard(HL.getCurrentCard()) << std::endl;

        // Let user bet or quit
        std::cout << std::endl
            << "Do you bet:\n 1. High (" << HL.getMultiplier('h') << ")\n"
            << " 2. Low (" << HL.getMultiplier('l') << ")\n"
            << " 3. Quit\n"
            << " Enter your choice: ";
        std::cin >> choice;

        // pull out next card
        HL.setNextCard(HL.getCard());
        std::cout << "The next card: " << HL.realCard(HL.getNextCard()) << std::endl;

        switch (choice)
        {
            // High bet
            case 1:
            {
                if (HL.compare(HL.getCurrentCard(), HL.getNextCard()) == 0)
                {
                    std::cout << "You win!" << std::endl;
                    HL.setPay(HL.getMultiplier('h'));
                    // Set the current card to nextCard if appropriate else get appropriate
card
                    if ((HL.getNextCard() != 1) && (HL.getNextCard() != 13))
                        HL.setCurrentCard(HL.getNextCard());
                    else
                    {
                        do
                        {
                            HL.setCurrentCard(HL.getCard());
                        } while ((HL.getCurrentCard() == 1) || (HL.getCurrentCard() == 13));
                    }
                }
                else if (HL.compare(HL.getCurrentCard(), HL.getNextCard()) == -1)
```

```
                std::cout << "It's a draw!" << std::endl;
            else
            {
                std::cout << "You lost!" << std::endl;
                system("PAUSE");
                return (-1 * HL.getBet());
            }
            system("PAUSE");
            break;
        }
        // Low bet
        case 2:
        {
            if (HL.compare(HL.getCurrentCard(), HL.getNextCard()) == 1)
            {
                std::cout << "You win!" << std::endl;
                HL.setPay(HL.getMultiplier('l'));
                // Set the current card to nextCard if appropriate else get appropriate
card
                if ((HL.getNextCard() != 1) && (HL.getNextCard() != 13))
                    HL.setCurrentCard(HL.getNextCard());
                else
                {
                    do
                    {
                        HL.setCurrentCard(HL.getCard());
                    } while ((HL.getCurrentCard() == 1) || (HL.getCurrentCard() == 13));
                }
            }
            else if (HL.compare(HL.getCurrentCard(), HL.getNextCard()) == -1)
                std::cout << "It's a draw!" << std::endl;
            else
            {
                std::cout << "You lost!" << std::endl;
                system("PAUSE");
                return (-1 * HL.getBet());
            }
            system("PAUSE");
            break;
        }
        // Quit
        case 3:
        {
            system("CLS");
            std::cout << "You won! Your pay is " << HL.getPay() << std::endl;
            return HL.getPay();
        }
        default:
        {
            std::cout << "Enter a valid choice.\n";
            system("PAUSE");
            break;
        }
    }

}
}
```

HighLow.cpp

```
/*
	Name:  Mike Apreza
	File:  HighLow.cpp
	Date:  03/07/2020
	Class: CIS 2220
*/

#include "HighLow.h"

double HighLow::getMultiplier(char type)
{
	switch (currentCard)
	{
	case 2:
	{
		if (type == 'l')
			return pays[1][0];
		else
			return pays[1][1];
	}
	case 3:
	{
		if (type == 'l')
			return pays[2][0];
		else
			return pays[2][1];
	}
	case 4:
	{
		if (type == 'l')
			return pays[3][0];
		else
			return pays[3][1];
	}
	case 5:
	{
		if (type == 'l')
			return pays[4][0];
		else
			return pays[4][1];
	}
	case 6:
	{
		if (type == 'l')
			return pays[5][0];
		else
			return pays[5][1];
	}
	case 7:
	{
		if (type == 'l')
			return pays[6][0];
		else
			return pays[6][1];
	}
	case 8:
	{
		if (type == 'l')
			return pays[7][0];
```

```
                else
                        return pays[7][1];
        }
        case 9:
        {
                if (type == 'l')
                        return pays[8][0];
                else
                        return pays[8][1];
        }
        case 10:
        {
                if (type == 'l')
                        return pays[9][0];
                else
                        return pays[9][1];
        }
        case 11:
        {
                if (type == 'l')
                        return pays[10][0];
                else
                        return pays[10][1];
        }
        case 12:
        {
                if (type == 'l')
                        return pays[11][0];
                else
                        return pays[11][1];
        }
        default:
                return 1.0;
        }
}

int HighLow::compare(int current, int next)
{
        if (current > next)
                return 1;
        else if (current < next)
                return 0;
        else
                return -1;
}

HighLow::HighLow(double bet)
{
        // get first card
        do
        {
                currentCard = getCard();
        } while ((currentCard == 1) || (currentCard == 13));
        // set pay = bet
        setBet(bet);
        pay = bet;
}

void HighLow::setCurrentCard(int card)
{ currentCard = card; }
```

```
int HighLow::getCurrentCard()
{ return currentCard; }

void HighLow::setNextCard(int card)
{ nextCard = card; }

int HighLow::getNextCard()
{ return nextCard; }

void HighLow::setPay(double multiplier)
{ pay *= multiplier; }

int HighLow::getPay()
{ return pay; }
```