Project 1

<Blackjack Game>

CIS-5 – 44188
Oscar Sandoval
April 17, 2017

Introduction

Blackjack

The game of Blackjack is a popular casino game where a player is faced against a dealer, who deals cards from a deck, and compete to see who can get the closest to 21, without going over it. The goal of the game is to see who can get the closest, and thus, the winner is decided based on who has the higher total score, so long as neither player goes over 21. If either the player or the dealer go over 21, the one who went over it will lose the game. This game is a comparative kind of game, where final scores are compared to one another to see who has the higher value within the range of 21, and whoever has the higher value will win the game.

The game itself implements strategy as well as luck, given that random cards are dealt each time, and there is no way to know which card will be dealt next. In the course of the game, the player will initially be given 2 cards, who may then decide whether to "hit" or "stand" based on the value of his cards. The game works in that the total value of all the cards in one's hand is added and based on that, the player must decide whether to ask for another card to be added to his hand, also known as "hit," or decide to keep his current hand, also known as "stand," and end his turn for the dealer to play. With each successive card a player decides to add to his hand, his risk of losing the game by going over 21 increases. However, if the player's total value in hand is a low number in the range of 21, then he may also lose against the dealer's hand. Once the player ends his turn, the dealer will then play as he, too, decides whether to "hit" or "stand" based on his score. Whoever gets the closest to 21, without going over it, at the end of both turns wins the game, and in the occasion that both the player and the dealer get the same score at the end, the dealer wins the game.

I decided to program this game because I have previously played the game of Blackjack before and have enjoyed the strategic side of the game, as well as the random assigning of cards that person is given. Due to that, I set out to program this game, thinking that it would be a moderately easy task. However, once having started it, I realized there were more obstacles than I had previously anticipated, and thus, the difficult I found in programming this game increased.

Summary

Project Size: About 360 lines

Number of variables: About 10

Number of Methods: 6

In programming this game, I tried to implement as many rules as I could that the original game has, but I was unable to include them all due to not yet knowing how to overcome certain obstacles. In this project, I had to learn some things that we have not yet learned in class, such as arrays, that make the task of programming a game such as this one easier than if no arrays were used. If allowed, I would improve this game for the final project by adding all of the rules seen in a casino game and improving the AI's ability to determine a better strategy to execute.

Program

```
/*
 * File:   main.cpp
 * Author: Oscar Sandoval
 * Created on April 17, 2017, 6:47 PM
 * Purpose: Game of Blackjack Version 3
 */

//System Libraries
#include <iostream> //Input - Output Library
#include <string>   //Needed to use strings
#include <cstdlib>  //For rand and srand
#include <ctime>    //Time for rand
#include <chrono>
#include <thread>
#include <iomanip>  //Format the output

using namespace std; //Name-space under which system libraries exist

//User Libraries

//Global Constants

//Function Prototypes
void menu();
int player(int*, int, short*, const string*, short*);
int dealer(int*, int, short*, const string*, short*);
int givCard(short*, const string*, short*);
int sum(int*);
```

```
void rules();

//Execution begins here
int main(int argc, char** argv) {
   //Array for each card name
   const string cards[13] = { "Ace", "Two", "Three", "Four", "Five", "Six", "Seven",
                  "Eight", "Nine", "Ten", "Jack", "Queen", "King" };

   //Array for value of each card
   short cardVal[13] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 10, 10, 10 };

   //Array for total number of each card value in deck
   short cardTot[13] = { 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4 };

   //Array for player's hand
   int pHand[11];

   //Array for dealer's hand
   int dHand[11];

   //Declare variables
   char choice; //Input number to choose option menu
   int pScore; //Total score of player based on his hand's total value
   int dScore; //Total score of dealer based on his hand's total value
   char again; //Input character to play again or not
   int pIndex = 0; //Initialize to 0 each time program loops
   int dIndex = 0; //Initialize to 0 each time program loops
   bool askAgn = false; //Loops again as long as boolean is false
   int win = 0; //Counts the wins of the player
```

```
int loss = 0; //Counts the losses of the player

do
{
   //Loop will reset all of the card values in array for player
   //and dealer back to zero at the start of each new game
   for (int i = 0; i < 11; i++)
   {
         pHand[i] = 0;   //Resets hand of player back to 0
         dHand[i] = 0;   //Resets hand of dealer back to 0
         cardTot[i] = 4; //Resets all cards of each value back to 4
   }

   askAgn = false; //Resets boolean back to false each time the program loops
   pIndex = 0; //Resets player's index each time program loops for array to start at the beginning
   dIndex = 0; //Resets dealer's index each time program loops for array to start at the beginning

   //Display the main menu
   menu();

   //Choose an option
   cout << "Type a number to choose your option: ";
   cin >> choice;

   switch (choice)
   {
      case '1':
         pScore = player(pHand, pIndex, cardTot, cards, cardVal); //Total score of player's hand returned from player function will be assigned to pScore
```

```
        //If player's score is less than 21, the game continues to dealer's turn
        if (pScore < 21)
        {
            dScore = dealer(dHand, dIndex, cardTot, cards, cardVal); //Total score of dealer's hand returned from dealer function will be assigned to dScore

            //Display the scores of the player and the dealer for comparison
            cout << "\nPlayer = " << pScore << endl;
            cout << "Dealer = " << dScore << endl;

            if (dScore > 21) //If dealer's score is greater than 21, player wins
            {
                cout << "\nYou win.\n\n" << endl;
                win++;
            }

            //Player's score must be greater than dealer's score to win
            if (pScore > dScore)
            {
                cout << "\nYou win.\n\n" << endl;
                win++;
            }
            //If dealer's score is greater than player's score, and less than 21, player loses
            else if (dScore > pScore && dScore <= 21)
            {
                cout << "\nYou lose.\n\n" << endl;
                loss++;
            }
```

```
    else if (pScore == dScore)
    {
        cout << "\nYou lose.\n\n" << endl;
        loss++;
    }
}
//If player's score is greater than 21, the game ends with a loss for the player
else if (pScore > 21)
{
    cout << "\nYou lose.\n\n" << endl;
    loss++;
}
//If player's score is equal to 21, player wins
else
{
    cout << "\nYou win\n\n" << endl;
    win++;
}

do //Ask player whether to play again or not
{
    cout << "Would you like to play again? (y/n)" << endl;
    cin >> again;

    if (again == 'n' || again == 'N') //If chosen not to play, the program will exit
    {
        //Output the game statistics to screen
        cout << fixed << setprecision(1) << showpoint;
        cout << "Exit the program." << endl << endl;
```

```
            cout << "Stats for this game were:" << endl;
            cout << "Total wins = " << win << endl;
            cout << "Total losses = " << loss << endl;
            cout << "Total games = " << win + loss << endl;
            cout << "Percentage of games won = " << static_cast<float>(win) /
                  (win + loss) * 100 << "%" << endl;
            cout << "Percentage of games lost = " << static_cast<float>(loss) /
                  (win + loss) * 100 << "%" << endl;
            exit(0);
         }
         else if (again == 'y' || again == 'Y') //If chosen to play, the program will run again
         {
            break;
         }
         else //If option is invalid, question will continue to be asked
         {
            cout << "Invalid option." << endl;
            askAgn = true;
         }
      } while (askAgn);
      break;
   case '2':
      rules(); //Display rules of Blackjack
      break;
   case '3':
      cout << "Exit the program." << endl;
      break;
   default:
      cout << "Not a valid option." << endl;
```

```
        }
    }while (choice != '3');

    return 0;
}

// Create menu for game
void menu()
{
    cout << "================================" << endl;
    cout << "          BLACKJACK           " << endl;
    cout << "================================" << endl << endl;
    cout << "Welcome to the game of Blackjack" << endl;
    cout << "What would you like to do?" << endl << endl;
    cout << "1. Play" << endl; //Option 1 plays the game
    cout << "2. Rules" << endl; //Option 2 displays the rules of the game
    cout << "3. Quit" << endl << endl; //Option 3 exits the program
}

// Function for player's turn
int player(int* pHand, int pIndex, short* cardTot, const string* cards, short* cardVal)
{
    bool askAgn = false; //Set boolean statement to false to continue loop
    int pTotal = 0; //Total sum of player's cards' values
    char choose;    //Choose whether to hit or stand

    //Deal the first two cards
    cout << "\n\nYour two cards are:" << endl;
    pHand[pIndex++] = givCard(cardTot, cards, cardVal);  //First card dealt will go into first space of array
```

```
   pHand[pIndex++] = givCard(cardTot, cards, cardVal);  //Second card dealt will go into second space of
array


   pTotal = sum(pHand); //Sum of player's hand returned from sum function will be assigned to pTotal


   if (pTotal < 21)
   {
      //Loop the question to hit or stand until player stands,
      //or wins or loses the game by hitting 21 or over 21
      do
      {
         pTotal = 0; //Reset pTotal back to 0 to give correct sum each time a new card is dealt


         cout << "\nWould you like to hit (h) or stand (s)?" << endl;
         cin >> choose; //Player inputs choice


         if (choose == 'h' || choose == 'H') //If chosen to hit, another card will be dealt
         {
            cout << "New card is:" << endl;
            pHand[pIndex++] = givCard(cardTot, cards, cardVal); //Card will go into next space in array


            pTotal = sum(pHand); //New sum will be calculated


            //Check to see if player has won or lost the game, else continue asking to hit or stand.
            if (pTotal > 21)
            {
               cout << "Your total is greater than 21." << endl;
               break; //Player has lost. Break out of loop and display a loss
            }
```

```
            else if (pTotal == 21)
            {
               cout << "Your total is equal to 21." << endl;
               break; //Player has won. Break out of loop and display a win
            }
            askAgn = true; //Loop will continue as long as player chooses to hit
         }
         else if (choose == 's' || choose == 'S') //If chosen to stand, player will end his turn
         {
            cout << "You have ended your turn." << endl;
            pTotal = sum(pHand); //Display sum of player's current hand
            askAgn = false; //Break out of question loop to continue the game
         }
         else
         {
            cout << "Invalid option." << endl;
            askAgn = true; //If option is invalid, continue to ask question until valid option is input
         }

      } while (askAgn); //Loop will continue as long as askAgain is true
   }

   return pTotal; //Returns player's total score back to main
}

// Function for dealer's turn
int dealer(int* dHand, int dIndex, short* cardTot, const string* cards, short* cardVal)
{
   int dTotal = 0; //Total sum of dealer's cards' values
```

```
   //Deal the first two cards
   cout << "\n\nDealer's cards are:" << endl;
   dHand[dIndex++] = givCard(cardTot, cards, cardVal);  //First card dealt will go into first space of array
   dHand[dIndex++] = givCard(cardTot, cards, cardVal);  //Second card dealt will go into second space of
array


   dTotal = sum(dHand); //Sum of dealer's hand returned from sum function will be assigned to dTotal


   do
   {
      if (dTotal < 17) //If total is less than 17, dealer has to hit.
      {
         dTotal = 0; //Reset dTotal back to 0 to give correct sum


         cout << "Dealer hits. New card is: " << endl;
         dHand[dIndex++] = givCard(cardTot, cards, cardVal);


         dTotal = sum(dHand);


         if (dTotal > 21)
         {
            cout << "Dealer lost." << endl;
         }
      }
      else if (dTotal >= 17) //If total is greater than or equal to 17, dealer has to stand.
      {
         cout << "The dealer stands." << endl;
      }
```

```
    } while (dTotal < 17); //Loop as long as score is less than 17
    return dTotal; //Returns dealer's total score back to main
}

// Deals cards to dealer and player and removes them from cards in deck
int givCard(short* cardTot, const string* cards, short* cardVal)
{
    int cIndex; //Card index used to create random cards and assign a name and value to each

    // NOTE: We wait 1000 milliseconds because fast CPUs
    // give the same number inside random.
    this_thread::sleep_for(chrono::milliseconds(1000));

    //Set the random number seed
    std::srand(static_cast<unsigned int>(time(0)));

    do
    {
        //Gives out a random card
        cIndex = rand() % 12;

        //Removes the card from the deck.
        cardTot[cIndex] -= 1;

        //Will continue to deal cards as long as there still are cards of that value
        if (cardTot[cIndex] > 0)
        {
            cout << cards[cIndex] << " (" << cardVal[cIndex] << ")" << endl;
        }
```

```
    } while (cardTot[cIndex] == 0);
    return cardVal[cIndex]; //Returns the value of each card back to dealer and player functions
}

// Sum for the value of all cards in your hand
int sum(int* hand)
{
    int total = 0;

    for (int i = 0; i < 11; i++)
    {
        total += hand[i]; //Adds the total amount of the value of all the cards in your hand
    }
    cout << "The total value of the cards is " << total << endl;

    return total; //Returns sum of all cards in hand to dealer and player functions
}

// Rules of Blackjack
void rules()
{
    cout << "\n\nThe rules of this game are as follows:" << endl << endl;
    cout << "There are two players in this game, you and the dealer." << endl;
    cout << "Each player will initially get two cards from a deck of 52 "
            "cards. " << endl;
    cout << "After getting the cards, you will be given a choice to hit or "
            "stand." << endl;
    cout << "If chosen to hit, you will be given another card in addition to "
            "those you had." << endl;
```

```
    cout << "If chosen to stand, you will keep your current hand, and your "
        "turn will end." << endl;
    cout << "Each card has a value corresponding to the number that is shown "
        "on them.\nThe special face cards of Jack, Queen, and King, each have a "
        "value of 10, " << endl;
    cout << "whereas the Ace has a value of 1." << endl;
    cout << "The goal of the game is to see who can get the closest to "
        "the number 21, with any combination of cards you have. " << endl;
    cout << "However, if you were to pass the number 21 with the added value "
        "of all of your cards, you will lose the game." << endl;
    cout << "Whoever gets the closest to 21 without going over it, will win "
        "the game." << endl;
    cout << "In the occasion that both players get the same score in the end, "
        "the dealer wins the game.\n\n" << endl;
}
```