# Project 2

<Blackjack>

Modified

CSC - 5 - 46090

Allende, Katelyn

7/31/2015

# Table of Contents

# Description

## Title: Blackjack Game

Blackjack is a popular casino game that is now played all around the world. In the game the aim of the player is to achieve a hand whose points total nearer to 21 than the dealer's hand without exceeding 21.

### Equipment:

The game of Blackjack is played with a traditional 52-card deck minus the jokers. Each of these cards are valued at the following:

- An Ace can equal either 1 or 11, whichever makes a better hand.
- Cards 2 from 9 are valued at their face value.
- The 10, Jack, Queen, and King are all valued at 10.

### Rules of the Game:

At the start of the game each player starts with two cards. One of the dealer's cards is hidden throughout the entire game until the end. Upon looking at their first two cards the player has two choices. The player can either 'Hit' and receive an additional card or 'Stay' to hold their total and terminate their turn. The dealer must hit until their cards total 17 or higher.

If the player's hand exceeds 21 they bust and the dealer automatically wins no matter the score of their hand. If the player's hand is closer to 21 without exceeding that amount they have beat the dealer. Finally, a blackjack is a total of 21 in their first two cards, which results in an automatic win.

### Betting System:

-If the player wins, their winnings are double the bet they made for that hand.

-If the player losses, they lose the amount they beat for that hand.

-If the hand ends in a tie, the player gains nothing, but gets the amount of money they bet for that hand back in their pot.

# Summary

Project Size: 389 lines

Number of Variables: 36

Number of Methods: 31

- This program is a modified game and took approximately a week.
- Modification of the program made the game more accurate to the real game (bet system).
- The difficulty of the project was moderate and made easier after learning how to apply functions.
- 31 concepts were utilized, however there were 2 constructs I was unable to incorporate.
    1. Sorting/Searching Arrays
       - Even though I know how to sort arrays, for this game of Blackjack it did not make logical sense for me to add it in I thought. I had the idea to have the player's hand sorted after each hit, but that would just confuse the player on which new card they received.
    2. 2D Arrays
       -Unsure how to implement these into my program. Guess the game of Battleship would have better for that!
- It was challenging to program my first game, but rewarding after it ran successfully.

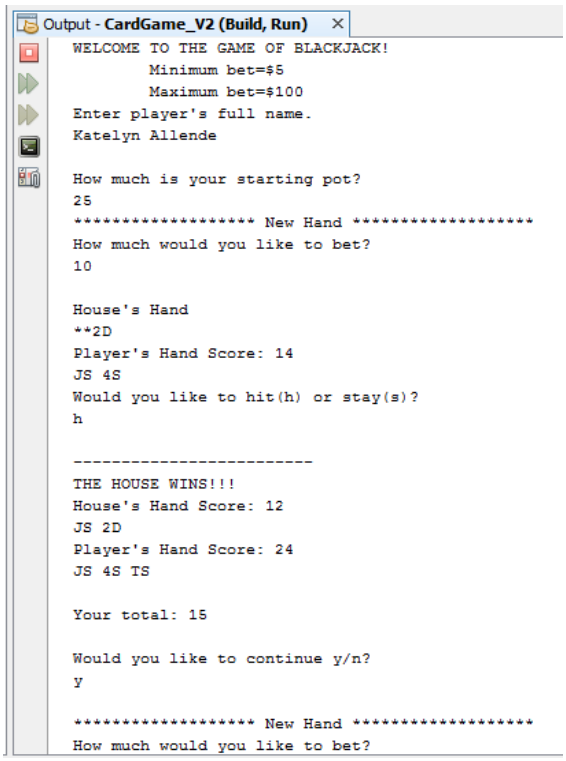
# Objective

To program a game using the constructs learned in class. The program allows the player to play the game of Blackjack against the dealer (computer) with the incorporation of bets.

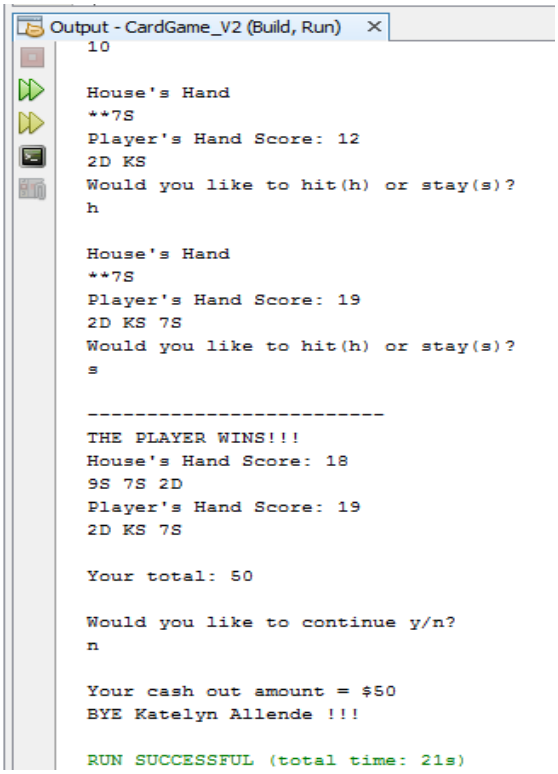
# Implementation

I pulled from a file to output the welcome heading. Used the random number generator to shuffle the array with the deck of cards. Used multiple functions to shuffle deck, print cards and hands, score hand, deliver next card, print final score and hand as well as the player's updated pot amount. In the main it allows the player to input their name, starting pot amount, bet amount, whether or not they would like to hit or stay and if they would like to continue playing. A combination of if-else statements, loops and calling of functions allows the program to output the winner based on their score. Throughout the main there are also various input validations to ensure the game is being played correctly.

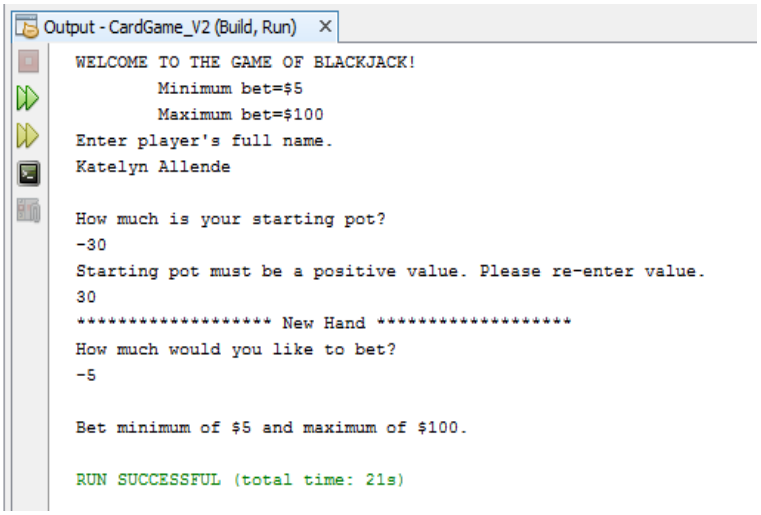
# Sample I/O



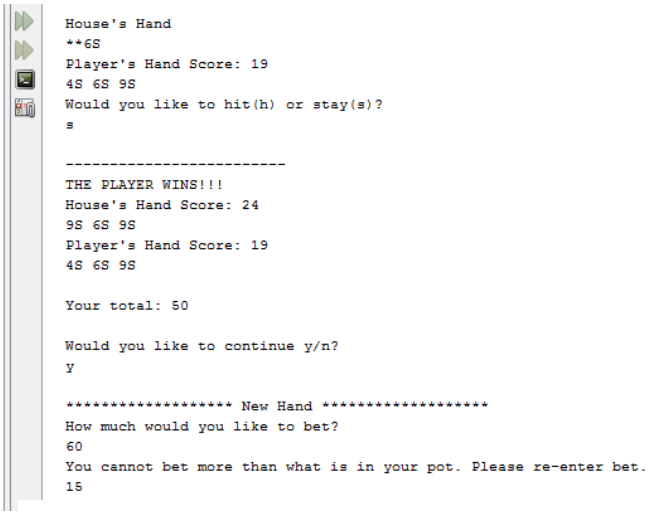

Shows a full hand and the ability to play again




Player no longer wishes to play. Outputs cash out amount and says bye.




Input Validation:
Bet Minimum/Maximum




Input Validation:
Bet Amount exceeds Pot Amount

# Variables

| Type | Variable Name | Description |
|---|---|---|
| **bool** | CrdsDlt[52] | Deck of Cards |
| **int** | hCrdCnt, pCrdCnt | House / Player Card Count |
| **int** | *HseHnd/PlyrHnd=new int [12] | Array of House/ Player Hand |
| **bool** | doAgain | Loop decision to play again or not |
| **char** | pChoice | Choice to Hit or Stay |
| **bool** | pHits | Player Hits |
| **int** | pScore, hScore | Player's / House's Score |
| **bool** | hBusts | House Bust |
| **char** | response | Decision to continue or not |
| **const int** | hCnt, pCnt | House/ Player Count |
| **int** | h | Hand |
| **const int** | t | Card Count |
| **int** | AceCnt | Ace Count |
| **int** | Score | Score of Hand |
| **int** | newCrd, newCard | New Card |
| **int** | iRank | Rank of card |
| **int** | r | Loop to score/print hand |
| **bool** | d | Decision for next card or not |
| **const int** | iNxtCrd | Next Card |
| **const int** | cRank, cSuit | Card Rank/ Suit |
| **char** | name | Player's first/ last name |
| **int** | Pot | Pot Amount |
| **int** | betAmnt | Bet Amount |
| **string** | line | Welcome Line |
| **int** | p | Card |
| **bool** | win | Player won hand |
| **float** | sAmnt | Original pot amount |
| **float** | bet | Amount bet by player |
| **float** | newPot | Updated Pot Amount |
| **int** | i | Loop to shuffle cards |

# Cross Reference

| Syntax/Keywords | Location |
|---|---|
| **System Libraries/ Math Library** | iostream, cstdlib, ctime, string, fstream |
| **void (functions)** | void shuffle, PrntCrd, PrntHnd, PrntSaH |
| **int (functions)** | int NxtCard, ScrHnd |
| **float (function)** | float NewPot |
| **pointers** | Functions: shuffle (CrdsDlt), PrntHnd (h), NxtCard (CrdsDlt), HseHnd, PlyrHnd |
| **Pass by reference** | Functions: PrntSaH (hCnt, pCnt) |
| **Arrays / Dynamic** | Functions: shuffle (CrdsDlt), ScrHnd (h)<br>Dynamic: PrntSaH (HseHnd, PlyrHnd) |
| **string** | line |
| **switch** | pChoice |
| **ifstream (file input)** | inFile |
| **Boolean** | CrdsDlt, doAgain, hBusts, d, win |
| **srand** | (time(0)) |
| **rand ()** | newCard |
| **do-while** | doAgain, NxtCard (d), pHits && pScore < 22 |
| **while** | (pHits&&pScore<22), (hScore<17), (AceCnt>0&&Score<12), getline (inFile,line), Pot<0, betAmnt>Pot |
| **if-else** | pChoice, pScore, hBusts, iRank, cRank, cSuit, Pot>0, betAmnt>=5 && betAmnt<=100, response |
| **if** | inFile.is_open, !CrdsDlt[newCard] |
| **for** | Functions: ScrHnd (r), PrntHnd (r), shuffle (i) |
| **true** | pHits, doAgain, d, NewPot |
| **false** | pHits, doAgain, d, CrdDlt[i], NewPot |
| **Relational and logical operators (==,>,<,&&,!,\|\|)** | Numerous locations throughout code |
| **Arithmetic operators (+, - , = , *)** | Numerous locations throughout code |
| **Increment operator (++)** | pCrdCnt, hCrdCnt, r, AceCnt, Score, i |
| **Decrement operator (--)** | AceCnt |
| **%** | iRank, newCard, cRank, cSuit |

| | |
|---|---|
| **Variable types** | Bool, int, const int, char, float, |
| **Arithmetic Operators Precedence** | Function NewPot (newPot=sAmnt+(bet*2)) |
| **Conditional Operator** | Function NewPot (win)?(newPot=sAmnt+(bet*2)):(newPot=sAmnt-bet); |
| **Nesting Loops** | Numerous locations throughout code |
| **Structures** | input |
| **void** | Functions: PrntSaH, PrntHnd, PrntCrd, shuffle |
| **return** | 0<br>Functions: ScrHnd (Score), NxtCard (newCard), NewPot(newPot) |

ALSO...

- Mathematical Statements
- Pass by Value

# Pseudo-Code

```
System Libraries
Structure input
Function Prototypes
Declare Variables
Welcome/Rules Heading pulled in from a file (instrct.dat)
Using structure output player's full name
Starting Pot Amount
      While (Pot is less than 0)
          Starting pot must be a positive value. Please re-enter value.
Do
          If (Pot is greater than 0)
          Set the random number seed
          Shuffle the cards
          Deal two cards to each player
          Heading for a new hand
          Bet Amount
              While (betAmnt is greater than Pot)
                    You cannot bet more than what is in your pot. Please re-enter bet.
          If (betAmnt is greater than or equal to 5 and betAmnt is less than or equal to 100)
     Do
          Display first two cards
          Ask player if they would like to hit or stay
             If  (pChoice equals equals 'h' or 's')
                    Switch Statement (pChoice)
                              case 'h':
                                        PlyrHnd[pCrdCnt] equals pull from NxtCard function
                                        Increment operator for pCrdCnt
                              case 's':
                                        Hit equals false
                              default:
                                        "Invalid input. Try Again."
               Else
                    "Invalid input. Try Again."

          Update score, check for bust, and see who won
          pScore equals pull from ScrHnd function  (PlyrHnd , pCrdCnt)
          While (pHits and pScore is less and 22)
                    If  pScore is greater than 21
                              "THE HOUSE WINS!!!"
                              Pull from PrntSaH function
                              Pot equals NewPot function (false, Pot, betAmnt)
                    Else if  pScore equals equals 21
                              "THE PLAYER WINS!!!"
                              Pull from PrntSaH function
                              Pot equals NewPot function (true, Pot, betAmnt)
                    Else
                              hScore equals pull from ScrHnd function (HseHnd, hCrdCnt)
```

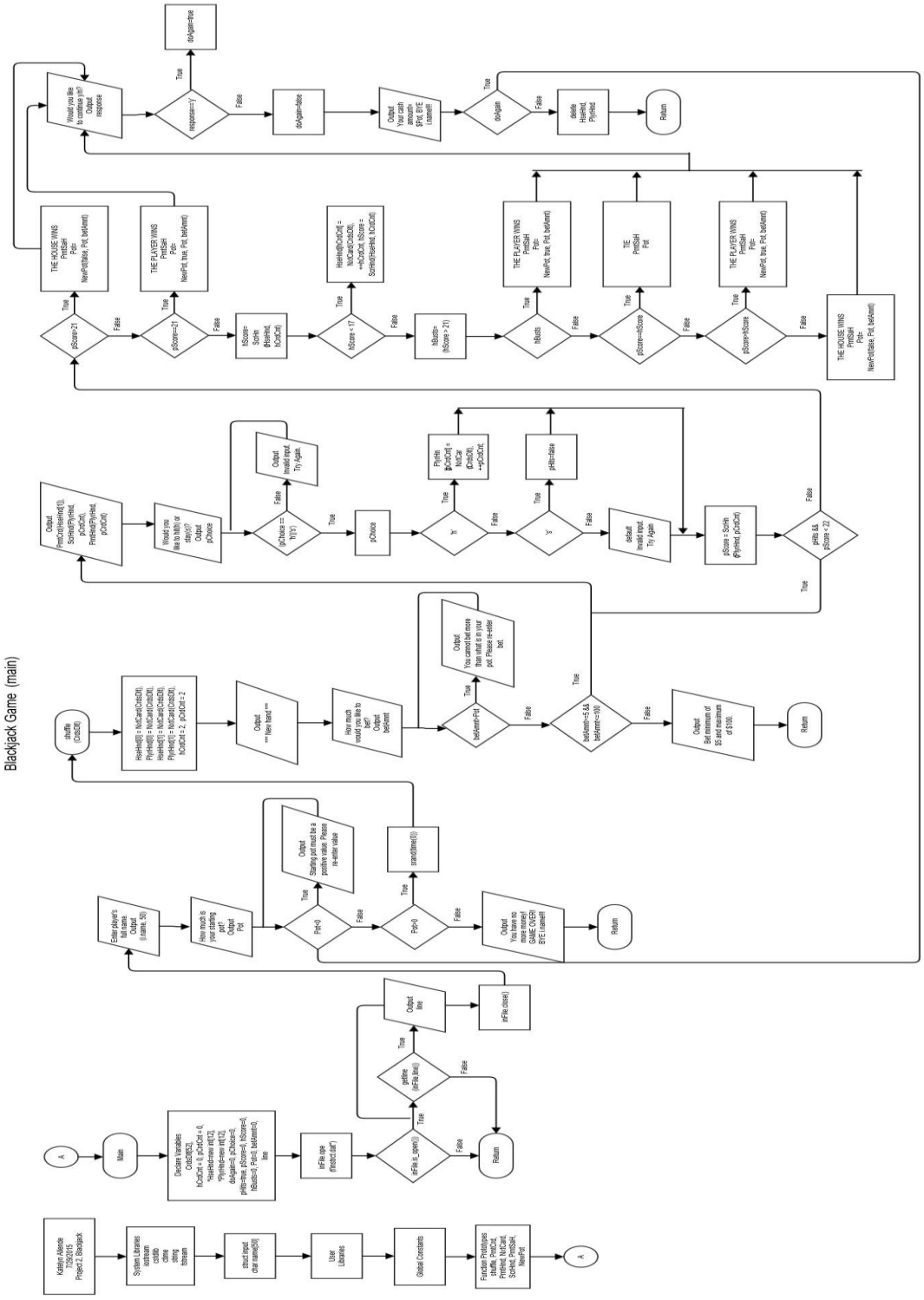
```
                    While hScore is less than 17
                        HseHnd [hCrdCnt] equals pull from NxtCard
                                                    function
                        Increment CrdCnt
                        hScore equals pull ScrHnd function (HseHnd, hCrdCnt)
                    hBusts equals hScore is less than 21
                    If hBusts
                        "THE PLAYER WINS!!!"
                        Pull from PrntSaH function
                        Pot equals NewPot function (true, Pot, betAmnt)
                    Else
                        If   pScore equals equals hScore
                            "TIE!!!"
                            Pull from PrntSaH function
                            Pot
                        Else if  pScore is greater than hScore
                            "THE PLAYER WINS!!!"
                            Pull from PrntSaH function
                            Pot equals NewPot function (true, Pot, betAmnt)
                        Else
                            "THE HOUSE WINS!!!"
                            Pull from PrntSaH function
                            Pot equals NewPot function (false, Pot, betAmnt)

            Else
                Bet minimum of $5 and maximum of $100.
            Else
                You have no more money! GAME OVER!
                Bye name (from structure) !!!
Prompt if they would like to continue
    If response equals equals 'y'
        doAgain equals true
    Else
        doAgain equals false
        Your cash out amount = $ Pot
        BYE name (from structure)!!!
While (doAgain)
delete HseHnd/PlyrHnd

Return

Void PrntSaH function
    ScrHnd function for both the Hse/Plyr Hnd and h/p Cnt

Int ScrHnd function
    Declare Variables
    For
        Increment r if r is less and t
        newCrd equals h[r]
        iRank equals newCrd mod 13
        If  iRank equals equals 0
```

```
                Increment AceCnt and Score
            Else if  iRank is less than 9
                Score equals score plus (iRank plus 1)
            Else
                Score equals score plus 10
        While AceCnt is greater than 0 and score is less than 12
            Decrement AceCnt
            Score equals score plus 10
        Return Score

Int NxtCard function
        Declare Variables
        Do
            newCard equals random number seed mod 52
            If CrdsDlt not [newCard]
            d equals false
        While (d)
        Return newCard

Void PrntHnd
        For
            Increment r if r is less than t
            iNxtCrd= h[r]
            Pull from PrntCrd function (iNxtCrd)

Void PrntCrd
        Declare Variables
        Print the rank of the card
        If cRank equals equals 0
            Output 'A'
        Else if cRank is less than 9
            cRank plus 1
        Else if cRank equals equals 9
            Output 'T'
         Else if cRank equals equals 10
            Output 'J'
        Else if cRank equals equals 11
            Output 'Q'
        Else
            Output 'K'

        Print the suit of the card
        If cSuit equals equals 0
            Output 'C'
        Else if cSuit equals equals 1
            Output 'D'
        Else if cSuit equals equals 2
            Output 'H'
        Else
            Output 'S'
```

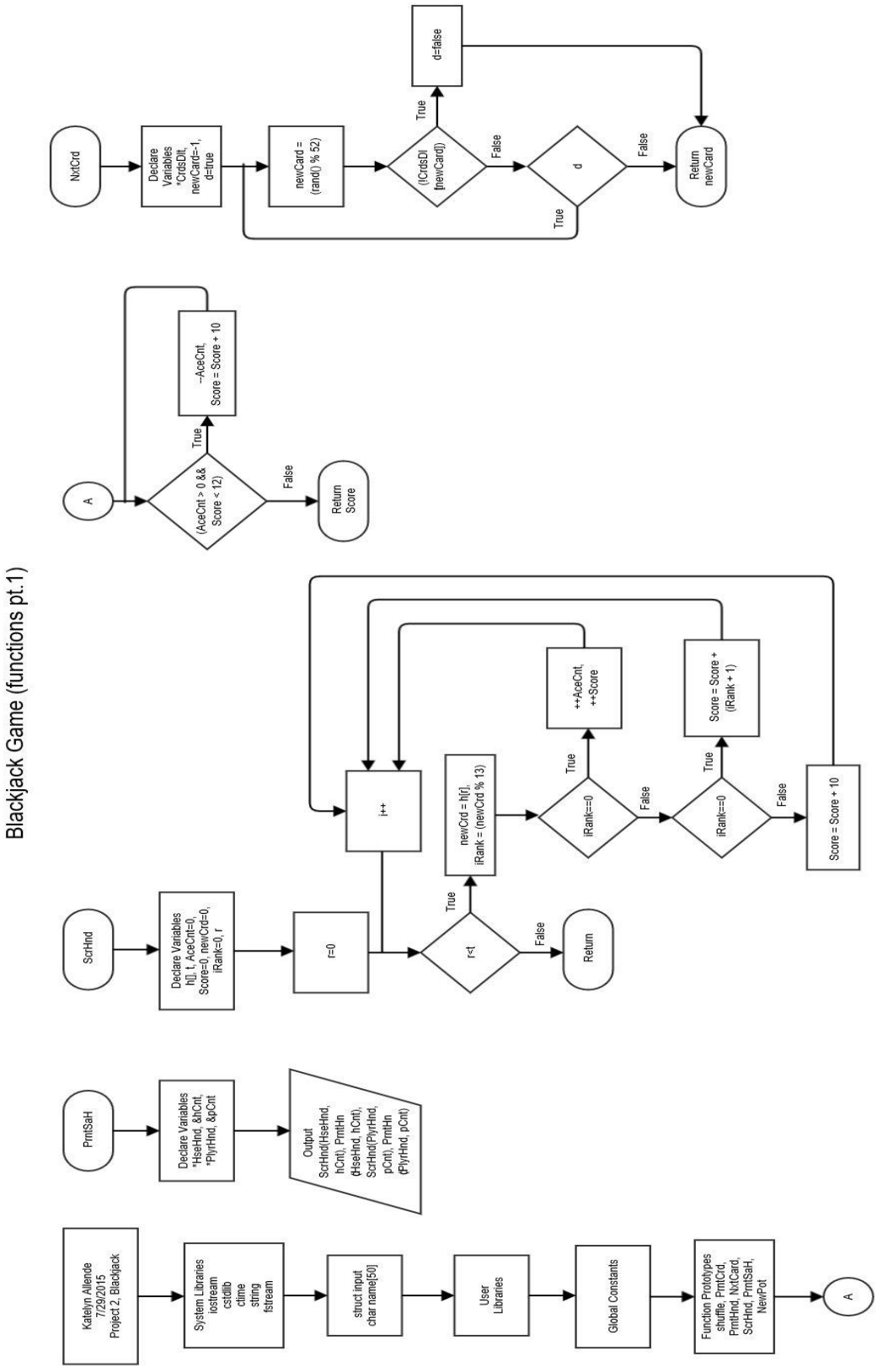
```
Void shuffle
        For
                Increment i if  i is less than 52
                CrdsDlt[i] equals  false
Float NewPot
        Declare Variables
        If (win)
                newPot equals sAmnt plus (bet times 2)
                return newPot
        Else
                newPot equals sAmnt minus bet
                return newPot
```

# Flowchart (main)

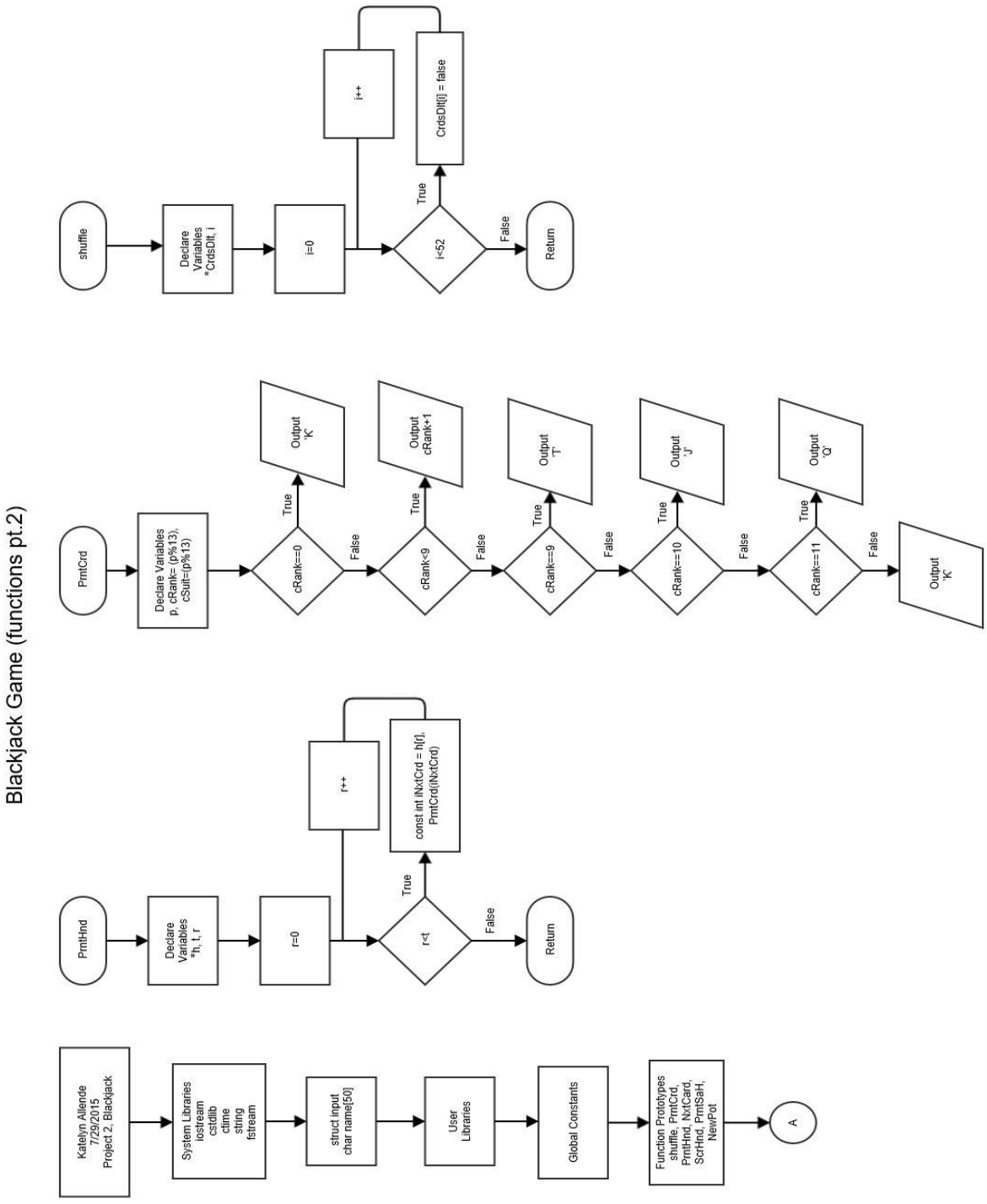
***flowcharts additionally in Project folder**



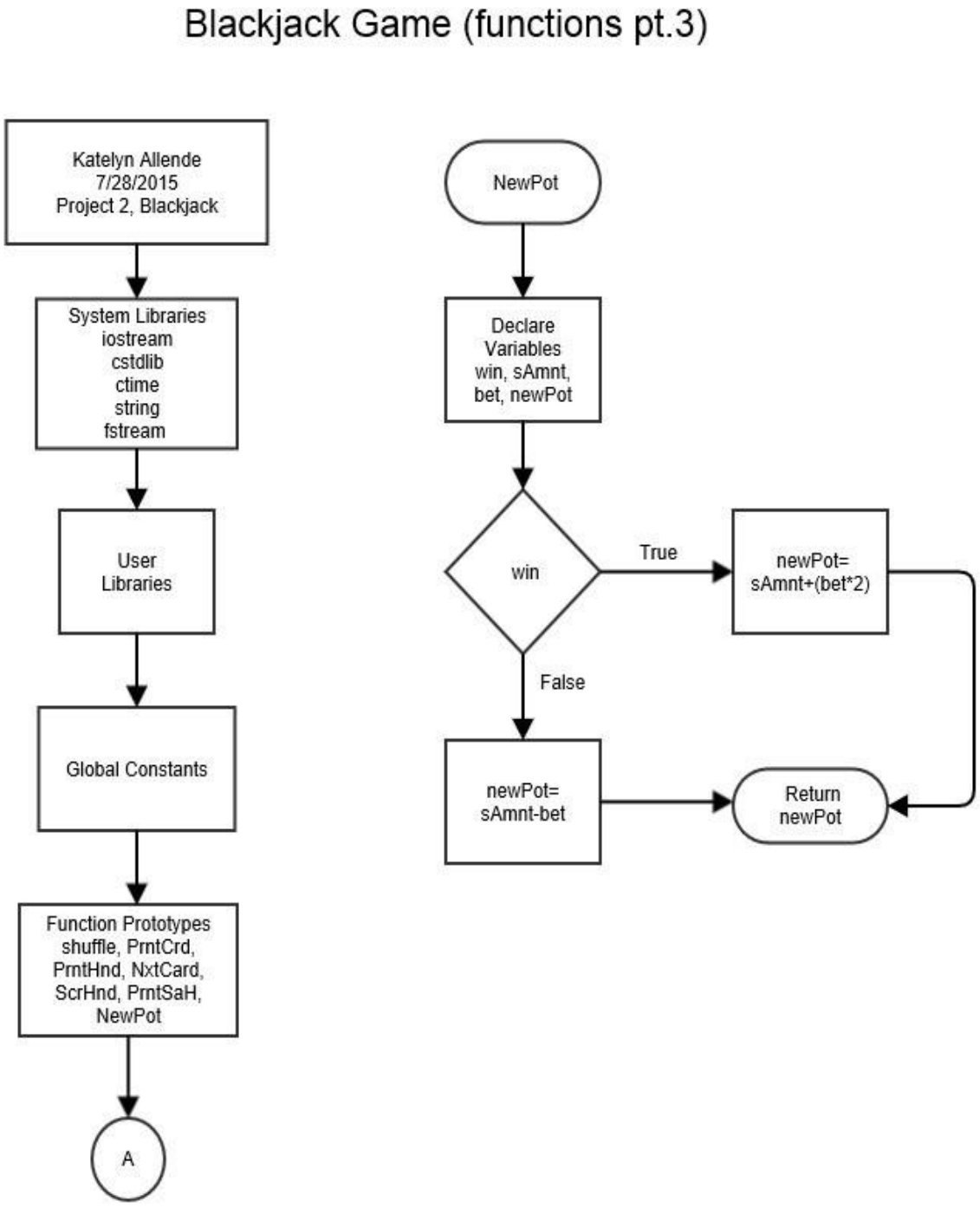
# Flowchart (functions pt. 1)



# Flowchart (functions pt. 2)



# Flowchart (functions pt. 3)



# Program

```
/*
 * File:   main.cpp
 * Author: Katelyn Allende
 * Created on July 26, 2015, 9:50 AM
 * Purpose: Project 2 - Create a game -
 *          Blackjack (modified)
 */

//System Libraries
#include <iostream>
#include <cstdlib>
#include <ctime>
#include <string>
#include <fstream>
using namespace std;

struct input {
   char name[50];
};
//User Libraries

//Global Constants

//Function Prototypes
void shuffle(bool *);
void PrntCrd(int);
void PrntHnd(int *, const int);
int NxtCard(bool *);
int ScrHnd(int [], const int);
void PrntSaH(int [], const int &, int [], const int &);
float NewPot (bool, float, float);

//Execution Begins Here!
int main(int argc, char** argv) {
   //Declare Variables
   bool CrdsDlt[52];               //Deck of Cards
   int hCrdCnt = 0, pCrdCnt = 0;     //House & Player Card Count
   int*HseHnd=new int[12];          //Dynamic Array of House Hand
   int*PlyrHnd=new int[12];         //Dynamic Array of Player Hand
   bool doAgain=0;                  //Decision to play again or not
   char pChoice=0;                  //Choice to Hit or Stay
   bool pHits = true;              //Player Hits
   int pScore=0, hScore=0;          //Player's & House's Score
```

```
bool hBusts=0;                  //House Bust
int Pot=0;                    //Pot Amount
int betAmnt=0;                  //Bet Amount

//Welcome to Blackjack
//Pulling from a file
string line;
ifstream inFile;
inFile.open("instrct.dat");
if (inFile.is_open()){

while (getline (inFile,line)){
   cout << line << endl;
}
inFile.close();
}

input i;   //Structure to output name
i.name;
cout << "Enter player's full name." << endl;
cin.getline(i.name, 50);

//Starting Pot Amount
cout << endl << "How much is your starting pot?" << endl;
cin >> Pot;
   //Input Validation
   while (Pot<0){
      cout << "Starting pot must be a positive value. Please re-enter value." << endl;
      cin >> Pot;
   }

   //Loop for each hand
   do {
      //Pot Amount Validation
      if (Pot>0) {
   //Set the random number seed
   srand(time(0));
   //Shuffle the cards
   shuffle(CrdsDlt);

   //Deal two cards to each player
   HseHnd[0] = NxtCard(CrdsDlt);
   PlyrHnd[0] = NxtCard(CrdsDlt);
   HseHnd[1] = NxtCard(CrdsDlt);
   PlyrHnd[1] = NxtCard(CrdsDlt);
   hCrdCnt = 2;
```

```
pCrdCnt = 2;

//Heading for a new hand
cout << "******************** New Hand ********************" << endl;

//Bet Amount
cout << "How much would you like to bet?" << endl;
cin >> betAmnt;
   //Input Validation
   while (betAmnt>Pot){
      cout << "You cannot bet more than what is in your pot. Please re-enter bet." << endl;
      cin >> betAmnt;
      }
cout << endl;
   //Input Validation
   if (betAmnt>=5 && betAmnt<=100) {

//Display first two cards
do {
   cout << "House's Hand" << endl;
   cout << "**";
   PrntCrd(HseHnd[1]);
   cout << endl;
   cout << "Player's Hand Score: " << ScrHnd(PlyrHnd, pCrdCnt) << endl;
   PrntHnd(PlyrHnd, pCrdCnt);

   //Ask player if they would like to hit or stay
   cout << "Would you like to hit(h) or stay(s)?" << endl;
   cin >> pChoice;
   if (pChoice == 'h'||'s'){
      switch(pChoice)
      {
         case 'h':
            PlyrHnd[pCrdCnt] = NxtCard(CrdsDlt);
            ++pCrdCnt;
            break;
         case 's':
            pHits=false;
            break;
         default:
            cout << endl << "Invalid input. Try Again." << endl;
      }
   }else {
      cout << "Invalid input. Try Again." << endl;
      }
   cout << endl;
```

```
//Update Score, Check for Bust, and See Who Wins
pScore = ScrHnd(PlyrHnd, pCrdCnt);
}while (pHits && pScore < 22);
   if (pScore > 21) {
      cout << "-------------------------" << endl;
      cout << "THE HOUSE WINS!!!" << endl;
      PrntSaH(HseHnd, hCrdCnt, PlyrHnd, pCrdCnt);
      Pot = NewPot(false, Pot, betAmnt);
      cout << "Your total: " << Pot << endl;
   }else if (pScore==21){
      cout << "-------------------------" << endl;
      cout << "THE PLAYER WINS!!!" << endl;
      PrntSaH(HseHnd, hCrdCnt, PlyrHnd, pCrdCnt);
      Pot = NewPot(true, Pot, betAmnt);
      cout << "Your total: " << Pot << endl;
   }else {
      hScore = ScrHnd(HseHnd, hCrdCnt);
      while (hScore < 17) {
         HseHnd[hCrdCnt] = NxtCard(CrdsDlt);
         ++hCrdCnt;
         hScore = ScrHnd(HseHnd, hCrdCnt);
      }
      hBusts = (hScore > 21);
      if (hBusts) {
         cout << "-------------------------" << endl;
         cout << "THE PLAYER WINS!!!" << endl;
         PrntSaH(HseHnd, hCrdCnt, PlyrHnd, pCrdCnt);
         Pot = NewPot(true, Pot, betAmnt);
         cout << "Your total: " << Pot << endl;
      }else {
         if (pScore == hScore) {
            cout << "-------------------------" << endl;
            cout << "TIE!!!" << endl;
            PrntSaH(HseHnd, hCrdCnt, PlyrHnd, pCrdCnt);
            cout << "Your total: " << Pot << endl;
         }else if (pScore > hScore) {
            cout << "-------------------------" << endl;
            cout << "THE PLAYER WINS!!!" << endl;
            PrntSaH(HseHnd, hCrdCnt, PlyrHnd, pCrdCnt);
            Pot = NewPot(true, Pot, betAmnt);
            cout << "Your total: " << Pot << endl;
         }else {
            cout << "-------------------------" << endl;
            cout << "THE HOUSE WINS!!!" << endl;
            PrntSaH(HseHnd, hCrdCnt, PlyrHnd, pCrdCnt);
```

```
                    Pot = NewPot(false, Pot, betAmnt);
                    cout << "Your total: " << Pot << endl;
                }
            }
        }
    } else {
        //Input Validation
        cout << "Bet minimum of $5 and maximum of $100." << endl;
        return 0;
    }
    } else {
        //No more money
        cout << "You have no more money! GAME OVER!" << endl;
        cin.ignore();
        cout << "BYE " << i.name << "!!!" << endl;
        return 0;
    }

  //Prompt if they would like to continue
  cout << endl << "Would you like to continue y/n?" << endl;
  char response;
  cin >> response;
  cout << endl;
  if (response == 'y') doAgain = true;
  else {
     doAgain = false;
     cout << "Your cash out amount = $" << Pot << endl;
     cin.ignore();
     cout << "BYE " << i.name << "!!!" << endl;
     }
  } while (doAgain);
  delete HseHnd;
  delete PlyrHnd;

 return 0;
}
/*****************************************************************
 ****************************** PrntSaH ***************************
 *****************************************************************
 * Purpose:     To print the score and hand.
 * Input:
 *         HseHnd    ->  House Hand
 *         hCnt      ->  House Card Count
 *         PlyrHnd   ->  Player Hand
 *         pCnt      ->  Player Card Count
 * Input-Output:
```

```
 *         ScrHnd    ->  Score Hand for House and Player
 *         PrntHnd   ->  Print Hand for House and Player
 ******************************************************************/
void PrntSaH(int *HseHnd, const int &hCnt, int *PlyrHnd, const int &pCnt) {
   //Output each score and the hand
   cout << "House's Hand Score: " << ScrHnd(HseHnd, hCnt) << endl;
   PrntHnd(HseHnd, hCnt);
   cout << "Player's Hand Score: " << ScrHnd(PlyrHnd, pCnt) << endl;
   PrntHnd(PlyrHnd, pCnt);
   cout << endl;
}
/******************************************************************
 ***************************** ScrHnd *****************************
 ******************************************************************
 * Purpose:     To determine the score of the hand.
 * Input:
 *         h       ->  Hand
 *         t       ->  Card Count
 * Output:
 *         score   ->  Score of hand
 ******************************************************************/
int ScrHnd(int h[], const int t) {
   //Declare Variables
   int AceCnt = 0;      //Ace Count
   int Score = 0;
   int newCrd=0;        //New Card
   int iRank=0;         //Rank of Card

   //For-loop to determine score of hand
   for (int r = 0; r < t; r++) {
      newCrd = h[r];
      iRank = (newCrd % 13);
      if (iRank == 0) {
         ++AceCnt;
         ++Score;
      } else if (iRank < 9) {
         Score = Score + (iRank + 1);
      } else {
         Score = Score + 10;
      }
   }
   while (AceCnt > 0 && Score < 12) {
      --AceCnt;
      Score = Score + 10;
   }
   return Score;
```

```
}
/*****************************************************************
 ****************************** NxtCrd ****************************
 *****************************************************************
 * Purpose:     To deal next card(s).
 * Input:
 *         CrdsDlt    ->  Deck of cards
 * Output:
 *         newCard    ->  Next card(s)
 *****************************************************************/
int NxtCard(bool *CrdsDlt) {
   //Declare Variable
   int newCard = -1;
   bool d = true;

   //Loop to get next card
   do {
      newCard = (rand() % 52);
      if (!CrdsDlt[newCard]) {
         d = false;
      }
   } while (d);
   return newCard;
}
/*****************************************************************
 ***************************** PrntHnd ****************************
 *****************************************************************
 * Purpose:     To print the card hand.
 * Input:
 *         h       ->  Hand
 *         t       ->  Card Count
 * Input-Output:
 *         On Screen
 *****************************************************************/
void PrntHnd(int *h, const int t) {
   //For-loop to print hand of cards
   for (int r = 0; r < t; r++) {
      const int iNxtCrd = h[r];
      PrntCrd(iNxtCrd);
      cout << " ";
   }
   cout << endl;
}
/*****************************************************************
 ***************************** PrntCrd ****************************
 *****************************************************************
```

```
 * Purpose:     To print the card.
 * Input:
 *          p       ->   Card
 * Input-Output:
 *          On Screen
 ******************************************************************************/
void PrntCrd(int p) {
   //Declare Variables
   const int cRank = (p % 13);     //Card Rank
   const int cSuit = (p % 13);     //Card Suit

   //Print the rank of the card
   if (cRank == 0) {
      cout << 'A';
   } else if (cRank < 9) {
      cout << (cRank + 1);
   } else if (cRank == 9) {
      cout << 'T';
   } else if (cRank == 10) {
      cout << 'J';
   } else if (cRank == 11) {
      cout << 'Q';
   } else {
      cout << 'K';
   }
   //Print the suit of the card
   if (cSuit == 0) {
      cout << 'C';
   } else if (cSuit == 1) {
      cout << 'D';
   } else if (cSuit == 2) {
      cout << 'H';
   } else {
      cout << 'S';
   }
}
/******************************************************************************
 ****************************** shuffle ***************************************
 ******************************************************************************
 * Purpose:     To shuffle the deck of cards.
 * Input:
 *          CrdsDlt        ->   Deck of cards
 * Input-Output:
 *          bool CrdsDlt   ->   Shuffled cards
 ******************************************************************************/
void shuffle(bool *CrdsDlt) {
```

```
   //For-loop to shuffle cards
   for (int i = 0; i < 52; i++) {
      CrdsDlt[i] = false;
   }
}
/*****************************************************************
 ***************************** NewPot *****************************
 *****************************************************************
 * Purpose:     To update total amount in pot.
 * Input:
 *         win       ->   Player won
 *         sAmnt     ->   Original pot amount
 *         bet       ->   Amount bet by player
 * Output:
 *         newPot    ->   New pot amount for player
 *****************************************************************/
float NewPot(bool win, float sAmnt, float bet) {
   //Declare Variables
   float newPot = 0.0f;       //Updated Pot Amount

   //Conditional Operator to calculate for new pot
   (win)?(newPot=sAmnt+(bet*2)):(newPot=sAmnt-bet);
   return newPot;
}
```

# References


Gaddis, Tony. *Starting out with C++ : From Control Structures through Objects*. Boston: Pearson, 2015. Print.

https://www.blackjackinfo.com/blackjack-rules/blackjack-basics/