Project 2

Blackjack

CSC-5-46332
Brandon Smith
07/31/2022

Introduction

Title: Blackjack

Blackjack, also known as 21, is a card game often played in casinos or with friends. The goal of the game is to get as close to 21 as you can without going over. In this version, you play against the dealer. The dealer must draw if they are under 17. If you have a higher score, or if the dealer busts while you don't, you win and receive double your bet.

Summary

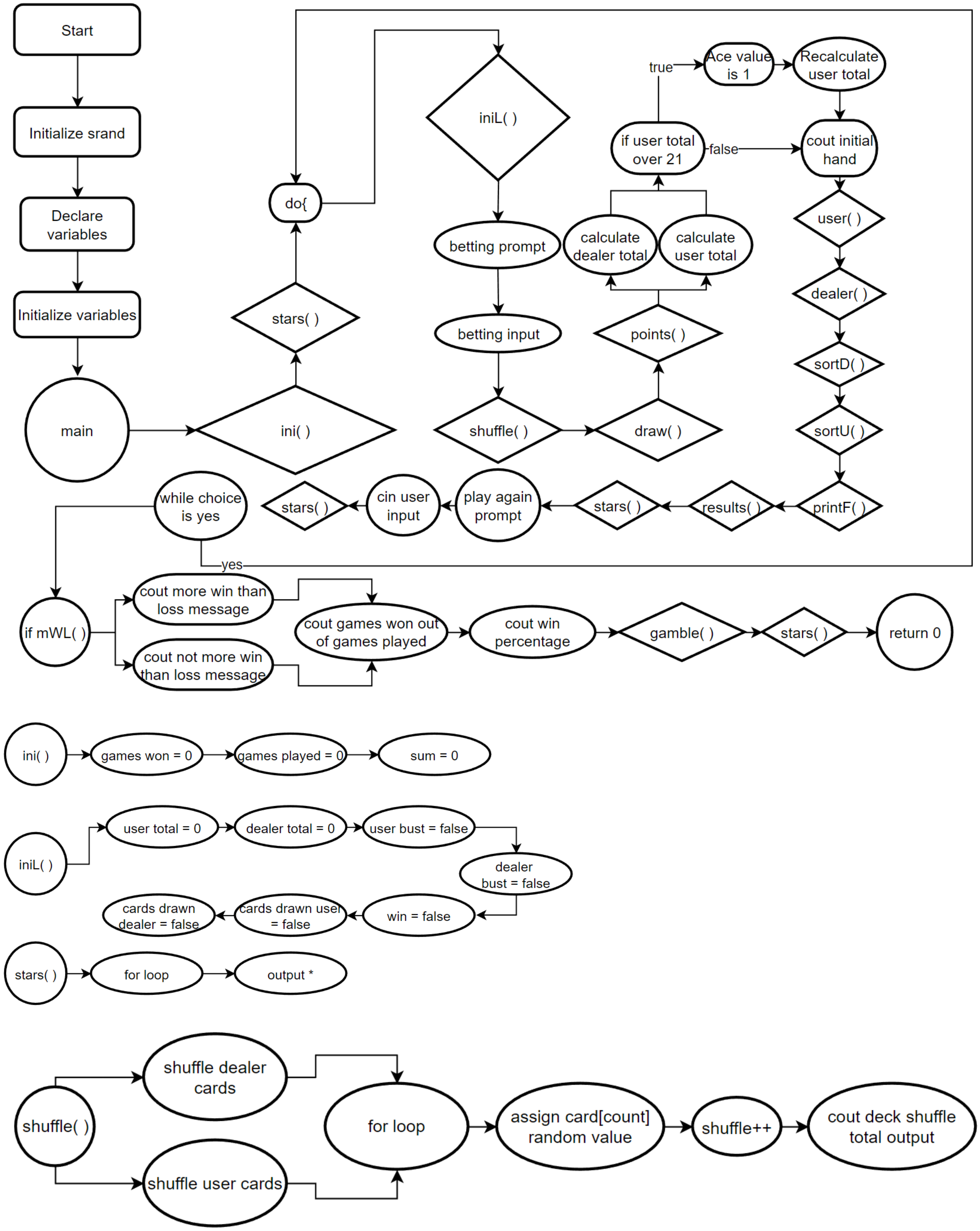
The program has 519 lines of code. It features several additional features to meet the various requirements required for the project.

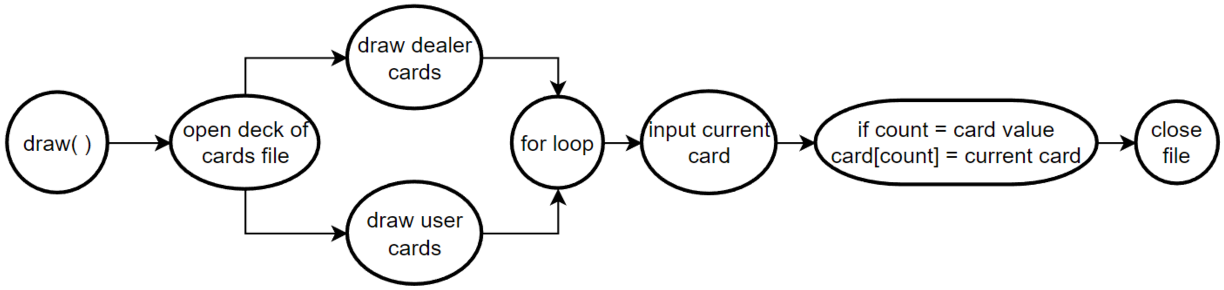
This project is much more complex compared to the previous project. This project had 4 separate versions, and the final version easily could have been split into two different ones because of the volume of change it underwent. There is only one bug that I am aware of, which is if a character is inputted for the bet resulting in an infinite loop. Aside from this error I am happy with how the program turned out. The program itself ended up being more bloated with additional features than I originally anticipated but they were necessary to meet the various requirements for this project.

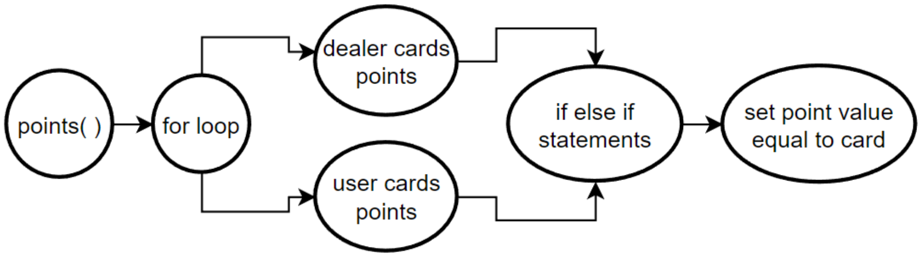
Description

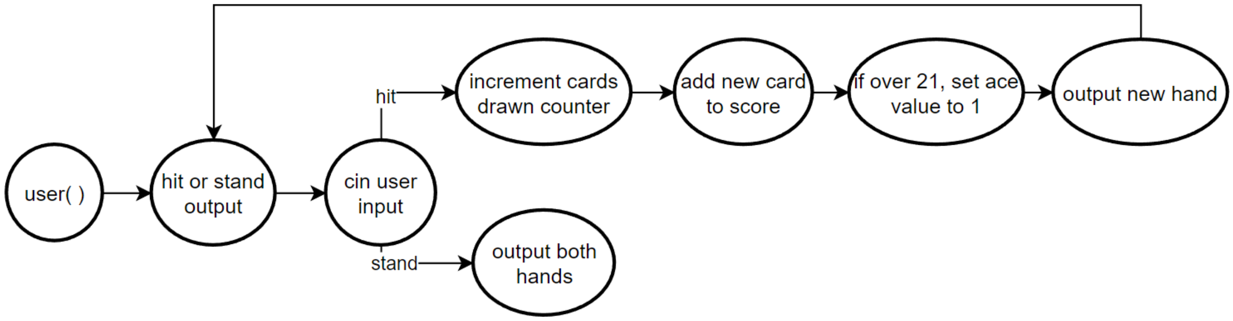
The primary focus of the program is to play blackjack. It also takes user bets and displays various statistics at the end of the game as well as the end of the program.

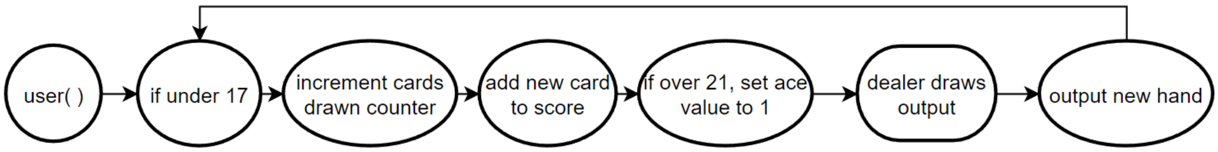
Flow Chart

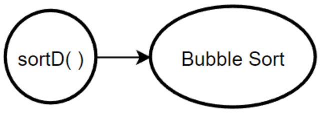

Start
Initialize srand
Declare variables
Initialize variables
main
ini( )
stars( )
do{
iniL( )
betting prompt
betting input
shuffle( )
draw( )
points( )
calculate dealer total
calculate user total
if user total over 21
true
Ace value is 1
Recalculate user total
false
cout initial hand
user( )
dealer( )
sortD( )
sortU( )
printF( )
results( )
stars( )
play again prompt
cin user input
stars( )
while choice is yes
yes
if mWL( )
cout more win than loss message
cout not more win than loss message
cout games won out of games played
cout win percentage
gamble( )
stars( )
return 0
ini( )
games won = 0
games played = 0
sum = 0
iniL( )
user total = 0
dealer total = 0
user bust = false
dealer bust = false
win = false
cards drawn user = false
cards drawn dealer = false
stars( )
for loop
output *
shuffle( )
shuffle dealer cards
shuffle user cards
for loop
assign card[count] random value
shuffle++
cout deck shuffle total output

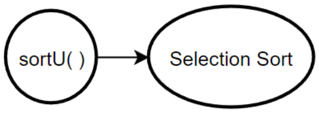

draw( )
open deck of cards file
draw dealer cards
draw user cards
for loop
input current card
if count = card value card[count] = current card
close file

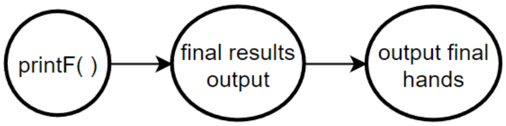

points( )
for loop
dealer cards points
user cards points
if else if statements
set point value equal to card

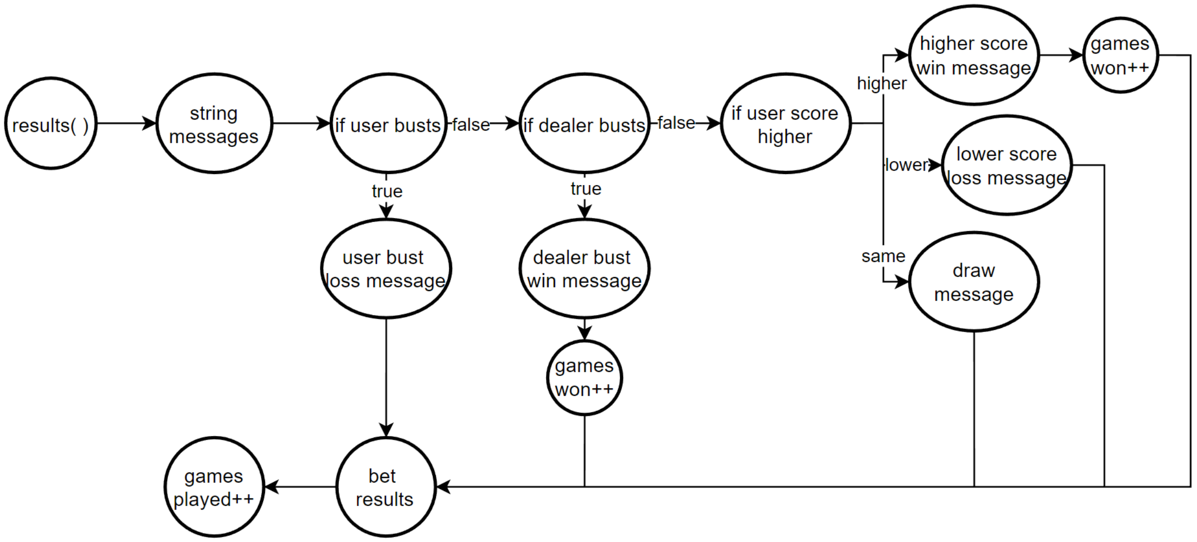

user( )
hit or stand output
cin user input
hit
increment cards drawn counter
add new card to score
if over 21, set ace value to 1
output new hand
stand
output both hands

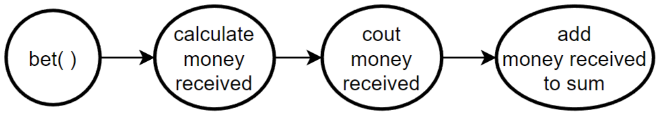

user( )
if under 17
increment cards drawn counter
add new card to score
if over 21, set ace value to 1
dealer draws output
output new hand

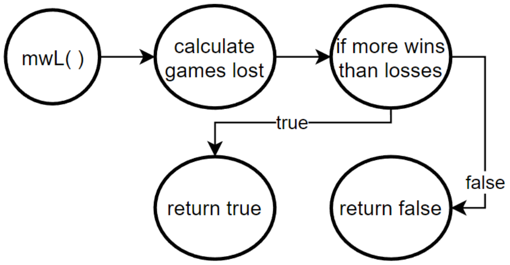

sortD( )
Bubble Sort

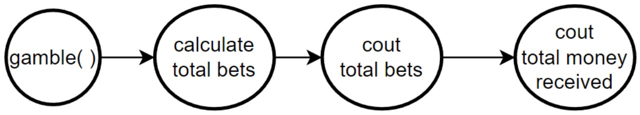

sortU( )
Selection Sort

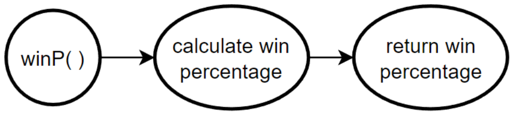

printF( )
final results output
output final hands


results( )
string messages
if user busts
false
if dealer busts
false
if user score higher
higher
higher score win message
games won++
lower
lower score loss message
same
draw message
true
user bust loss message
true
dealer bust win message
games won++
games played++
bet results


bet( )
calculate money received
cout money received
add money received to sum


mwL( )
calculate games lost
if more wins than losses
true
return true
false
return false


gamble( )
calculate total bets
cout total bets
cout total money received


winP( )
calculate win percentage
return win percentage

Pseudo Code

```
Initialize scrad
Declare variables
Initialize file parameters
Initialize variables

Start gameplay
Call initialize variables function
Call print stars function

Do{
    Call initialize loop variables function

    Cout betting input prompt
    Cin betting input
    While loop for input validation
    Cout invalid amount message
    Cin new betting input

    Call shuffle cards function
    Call draw cards function
    Call assign points function

    Output initial hands
    Calculate user total
    Calculate dealer total

    If user total is over 21
    For loop
    Set ace point value to 1
    Recalculate user total

    Cout formatting
    Cout dealers first card and point value
    Cout users hand and total points
    Cout formatting
    Call user turn function
    Call dealer turn function
    Call sort dealer hand function
    Call sort user hand function

    Call print stars function
    Cout play again prompt
    Cin play again input
    Call print stars function
}while user wants to play again

Cout if user won or lost more
    If win loss ratio function is false
        Cout more loss message
    If win loss ratio function is true
        Cout more win message
Cout games won out of games played
Cout win percentage
    Calculate win percentage

Call gamble results function
Call print stars function

Return 0;

Functions:
Initialize variables function
    Set default games won to 0
    Set default games played to 0
    Set default betting sum to 0

Initialize loop variables function
Set default user total to 0
Set default dealer total to 0
Set default user bust to false
Set default dealer bust to false
Set default win to false
Set default cards drawn by user to 2
Set default cards drawn by dealer to 2
```

```
Print star function
For loop
Cout *
Endl

Shuffle cards function
Static integer cards shuffled
For loop to shuffle user cards
        Calculate random values
For loop to shuffle dealer cards
        Calculate random values
Increment cards shuffled counter
Cout cards have been shuffled a total of _ times

Draw cards function
Input deck of cards file
Open deck of cards file
Loop to input cards
Input current card value from file
For loop to set user cards
        If current card value same as user card, set value
For loop to set dealer cards
        If current card value same as dealer card, set value
Close deck of cards file

Set point values function
For loop to assign user card points
        If card value is certain card, assign cards value
For loop to assign dealer card points
        If card value is certain card, assign cards value

User turn function
Cout hit or stand prompt
Cin hit or stand input
Validate user choice
        If invalid, cout invalid input message
        Cin new choice
```

```
Switch hit or stand
Case Hit
do{
    Add one to user cards drawn
    Add new card to user total
    If user if over 21
        Set ace value to 1 instead of 11
        Recalculate user score

    Cout formatting
    Cout dealers first card and points
    Cout users score
    Cout users cards
    Cout formatting

    If user does not bust
        Sets user bust to false
    If user busts
        Set user bust to true
        Exit program

    If user is under 21
        Cout hit or stand output
        Cin hit or stand input
        Validate user choice
            If invalid, cout invalid input message
            Cin new user input
    Else auto stand for user
}while hit or stand input is hit

Case Stand
Cout formatting
Cout dealers score
Cout dealers cards
Cout users score
Cout users cards
Cout formatting
```

```
Dealers turn function
If dealer is dealer is under 17 and user did not bust
    Do{
        Increment dealer cards drawn
        Add new card to dealer total
        Cout dealer draws output
        If dealer total is over 21
            Set ace value to 1 instead of 11
            Recalculate dealer score
        Cout formatting
        Cout dealers score
        Cout dealers cards
        Cout users score
        Cout users cards
    }while dealer is under 17
If dealer is under 21
    Set dealer bust to false
If dealer is over 21
    Set dealer bust to true

Sort dealers hand function
    Bubble sort cards

Sort users hand function
    Selection sort cards

Print final results function
Cout final results output
Cout formatting
Cout dealers score
Cout dealers cards
Cout users score
Cout users hand
Cout formatting
```

```
Win loss results function
String win loss messages

If user did not bust
    If dealer bust
        Cout dealer bust win message
        Increment games won message
        Call betting results function
    If dealer did not bust
        If user score is higher
            Cout higher score win message
            Increment games won message
            Call betting results function
        If user score is lower
            Cout lower score win message
            Call betting results function
        If user score is same
            Cout draw message
            Call betting results function
If user did bust
    Cout user bust loss message
    Call betting results function

Increment games played counter

Betting Results Function
Cout received amount
    Amount = bet*payout (0 loss, 1 draw, 2 win)
    Add amount to sum

Total Betting Results function
Loop for total bets value
    Add current bet to total
Cout total bets output
Cout sum won output
```

```
More wins or losses function
Calculate games lost
If more games lost
    Set status to false
If more games won
    Set status to true
Return status

Win percentage function
Calculate win percentage
Return percentage
```

Cross Reference from Project 1

Chapter	Section	Topic	Where Line#s	Pts	Notes
2	2	cout	70, 74, 93-98, 108-109, 115-116, 118-120, 151, 169, 239, 243, 266-270, 274-275, 278-279, 288, 292, 302-306, 310-311, 313-314, 326, 340-342, 346-347, 349-351, 355-356, 358-359, 421-424, 428-429, 431-433, 437-438, 440-441, 456, 464, 470, 475, 482, 490, 501-502		
	3	libraries	9-16	5	iostream, iomanip, cmath, cstdlib, fstream, string, ctime
	4	variables/literals	47-62, 158, 175-177, 236-237, 369-371, 396-397, 447-451, 469, 507-508, 516		No variables in global area, failed project!
	5	Identifiers	47-62, 158, 175-177, 236-237, 369-371, 396-397, 447-451, 469, 507-508, 516		
	6	Integers	47-50, 56, 370, 396, 508	1	
	7	Characters	60, 236-237	1	
	8	Strings	54-55, 176-177, 371, 397, 447-451	1	
	9	Floats No Doubles	58-59, 496, 516	1	Using doubles will fail the project, floats OK!
	10	Bools	61, 140-142, 362-363, 507, 509-510	1	
	11	Sizeof*****			
	12	Variables 7 or less	47-62, 158, 175-177, 236-237, 369-371, 396-397, 447-451, 469, 507-508, 516		All variables <= 7 characters
	13	Scope ***** No Global Variables			
	14	Arithmetic operators	82-83, 161, 166, 252, 325, 490-491, 499, 508, 517		

	15	Comments 20%+	60% of lines	2	Model as pseudo code
	16	Named Constants	47-49		All Local, only Conversions/Physics/Math in Global area
	17	Programming Style***** Emulate			Emulate style in book/in class repository
3	1	cin	71, 110, 240, 244, 289		
	2	Math Expression	82-83, 161, 166, 252, 325, 490-491, 499, 508, 517		
	3	Mixing data types ****			
	4	Overflow/Underflow ****			
	5	Type Casting	118-119	1	
	6	Multiple assignment *****			
	7	Formatting output	95, 97, 121, 268, 270, 275, 304, 306, 311, 342, 347, 351, 356, 424, 428-429, 437-438, 490, 501-502	1	
	8	Strings	54-55, 176-177, 371, 397, 447-451	1	
	9	Math Library		1	All libraries included have to be used
	10	Hand tracing ******			
4	1	Relational Operators	72, 85, 159, 164, 182, 185, 187, 192, 194, 206, 219, 253, 255, 257, 260, 271, 280-281, 286, 307, 321, 327, 329, 334, 343, 352, 360, 362-363, 375, 377, 398, 403, 405, 425, 434, 462, 468, 509-510		
	2	if	85, 187, 194, 253, 273, 280, 281, 321, 362, 363, 509, 510	1	Independent if
	4	If-else	115-116, 286-296, 452-480	1	

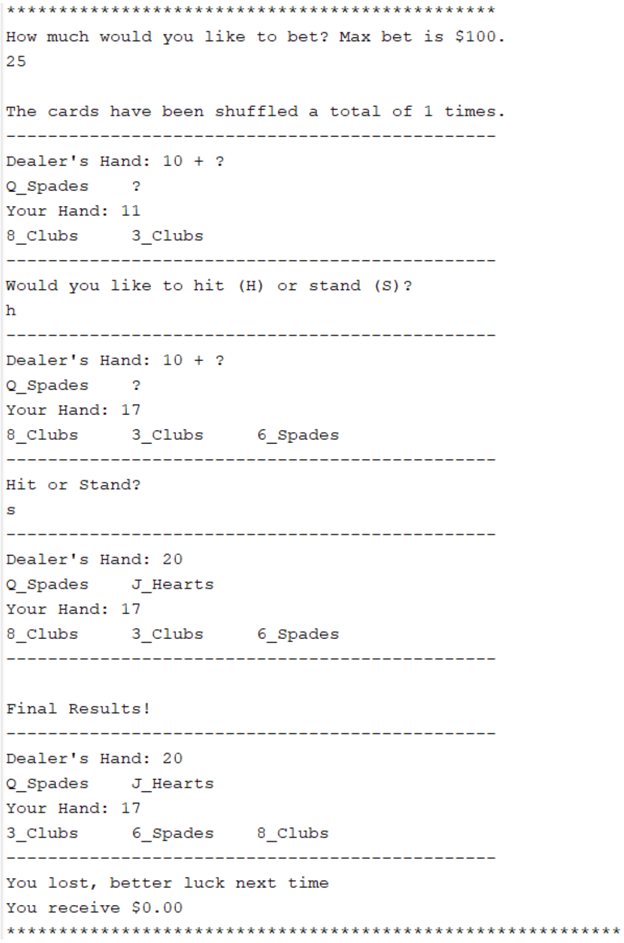
	5	Nesting	87, 89, 257, 327, 345, 377, 405, 427, 436, 454, 460, 468, 473	1	
	6	If-else-if	208-217, 221-230, 462-473	1	
	7	Flags *****			
	8	Logical operators	72, 89, 112, 208-216, 221-229, 241, 257, 273, 290, 345, 354, 427, 436, 452	1	
	11	Validating user input	72-76, 241-245, 290-295	1	
	13	Conditional Operator		1	
	14	Switch	246-315	1	
5	1	Increment/Decrement	168, 251, 324, 457, 465, 485	1	
	2	While	72-76, 241-245, 290-295	1	
	5	Do-while	67-112, 250-297, 323-360, 373-390	1	
	6	For loop	87, 149, 159, 164, 182, 185, 192, 206, 219, 255, 260, 271, 307, 329, 334, 343, 352, 375, 398, 403, 425, 434, 497	1	
	11	Files input/output both	172-202	2	
	12	No breaks in loops ******			Failed Project if included

Cross Reference from Project 2

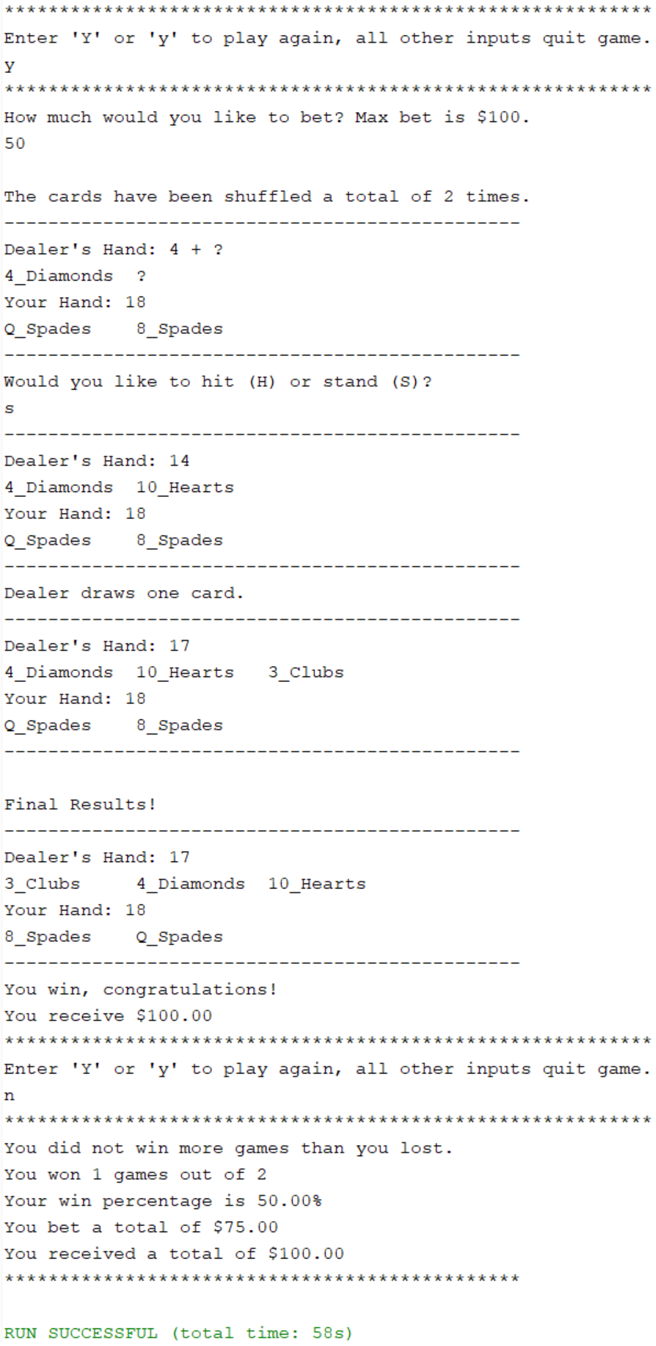
Chapter	Section	Topic	Where Line#s	Pts	Notes
6		Functions	23-39, 65, 66, 68, 78-80, 100-105, 107, 111, 115, 121, 122, 123		
	3	Function Prototypes	128-519	4	Always use prototypes
	5	Pass by Value	156, 172, 204, 234, 318, 366, 393, 419, 444, 488, 494, 505, 514	4	
	8	return	505, 512, 514, 518	4	A value from a function
	9	returning boolean	505, 512	4	
	10	Global Variables		XXX	Do not use global variables -100 pts
	11	static variables	158	4	
	12	defaulted arguments	26, 66, 123, 147-153	4	
	13	pass by reference	129, 136, 156, 172, 204, 234, 318, 366, 393, 419, 444, 488	4	
	14	overloading	107, 111	5	
	15	exit() function	284	4	
7		Arrays	50-55		
	1-6	Single Dimensioned Arrays	185-198, 208-217	3	
	7	Parallel Arrays	50-53, 52-55, 206-231	2	
	8	Single Dimensioned as Function Arguments	94-97, 267-270, 303-313, 341-358, 423-440	2	
	9	2 Dimensioned Arrays		2	Emulate style in book/in class repository
	12	STL Vectors	62	2	
		Passing Arrays to and from Functions	156, 172, 204, 234, 318, 366, 393, 419, 444	5	
		Passing Vectors to and from Functions	444, 488, 494	5	

8		Searching and Sorting Arrays	172-202, 204-232, 366-391, 393-417		
	3	Bubble Sort	366-391	4	
	3	Selection Sort	393-417	4	
	1	Linear or Binary Search	172-202, 204-232	4	

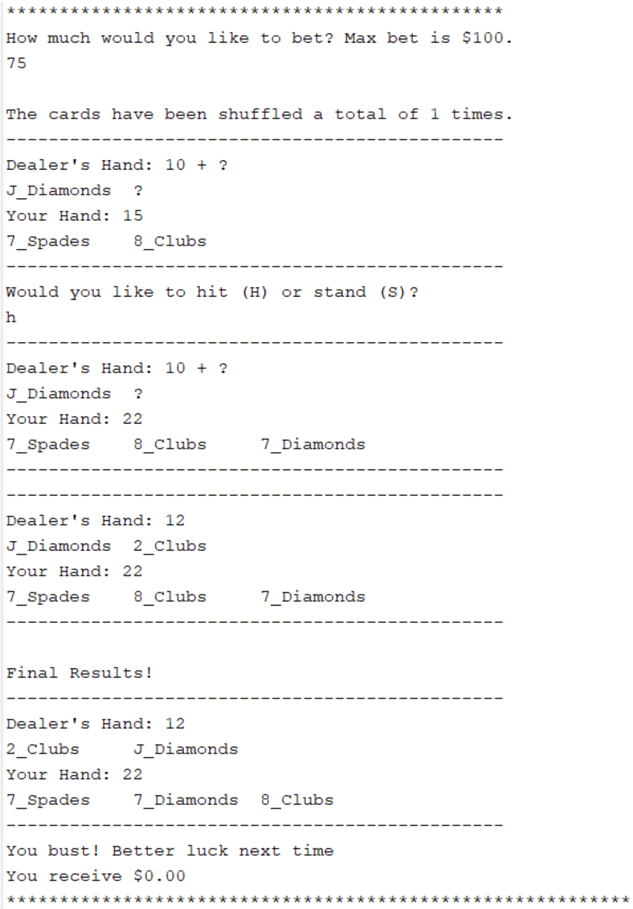
Proof of Working Code



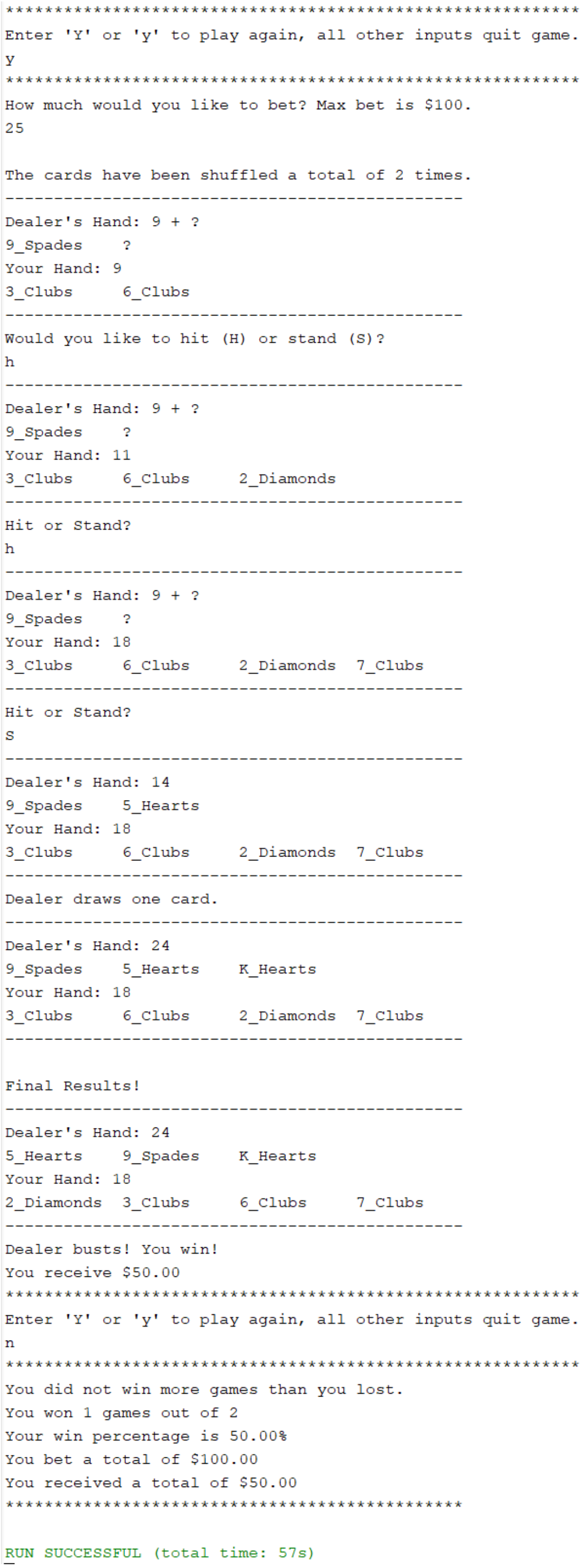
```
***********************************************
How much would you like to bet? Max bet is $100.
25

The cards have been shuffled a total of 1 times.
-----------------------------------------------
Dealer's Hand: 10 + ?
Q_Spades    ?
Your Hand: 11
8_Clubs     3_Clubs
-----------------------------------------------
Would you like to hit (H) or stand (S)?
h
-----------------------------------------------
Dealer's Hand: 10 + ?
Q_Spades    ?
Your Hand: 17
8_Clubs     3_Clubs     6_Spades
-----------------------------------------------
Hit or Stand?
s
-----------------------------------------------
Dealer's Hand: 20
Q_Spades    J_Hearts
Your Hand: 17
8_Clubs     3_Clubs     6_Spades
-----------------------------------------------

Final Results!
-----------------------------------------------
Dealer's Hand: 20
Q_Spades    J_Hearts
Your Hand: 17
3_Clubs     6_Spades    8_Clubs
-----------------------------------------------
You lost, better luck next time
You receive $0.00
***********************************************************
```



```
***********************************************************
Enter 'Y' or 'y' to play again, all other inputs quit game.
y
***********************************************************
How much would you like to bet? Max bet is $100.
50

The cards have been shuffled a total of 2 times.
-----------------------------------------------
Dealer's Hand: 4 + ?
4_Diamonds  ?
Your Hand: 18
Q_Spades    8_Spades
-----------------------------------------------
Would you like to hit (H) or stand (S)?
s
-----------------------------------------------
Dealer's Hand: 14
4_Diamonds  10_Hearts
Your Hand: 18
Q_Spades    8_Spades
-----------------------------------------------
Dealer draws one card.
-----------------------------------------------
Dealer's Hand: 17
4_Diamonds  10_Hearts   3_Clubs
Your Hand: 18
Q_Spades    8_Spades
-----------------------------------------------

Final Results!
-----------------------------------------------
Dealer's Hand: 17
3_Clubs     4_Diamonds  10_Hearts
Your Hand: 18
8_Spades    Q_Spades
-----------------------------------------------
You win, congratulations!
You receive $100.00
***********************************************************
Enter 'Y' or 'y' to play again, all other inputs quit game.
n
***********************************************************
You did not win more games than you lost.
You won 1 games out of 2
Your win percentage is 50.00%
You bet a total of $75.00
You received a total of $100.00
***********************************************

RUN SUCCESSFUL (total time: 58s)
```



```
***********************************************
How much would you like to bet? Max bet is $100.
75

The cards have been shuffled a total of 1 times.
-----------------------------------------------
Dealer's Hand: 10 + ?
J_Diamonds  ?
Your Hand: 15
7_Spades    8_Clubs
-----------------------------------------------
Would you like to hit (H) or stand (S)?
h
-----------------------------------------------
Dealer's Hand: 10 + ?
J_Diamonds  ?
Your Hand: 22
7_Spades    8_Clubs     7_Diamonds
-----------------------------------------------
-----------------------------------------------
Dealer's Hand: 12
J_Diamonds  2_Clubs
Your Hand: 22
7_Spades    8_Clubs     7_Diamonds
-----------------------------------------------

Final Results!
-----------------------------------------------
Dealer's Hand: 12
2_Clubs     J_Diamonds
Your Hand: 22
7_Spades    7_Diamonds  8_Clubs
-----------------------------------------------
You bust! Better luck next time
You receive $0.00
***********************************************************
```



```
***********************************************************
Enter 'Y' or 'y' to play again, all other inputs quit game.
y
***********************************************************
How much would you like to bet? Max bet is $100.
25

The cards have been shuffled a total of 2 times.
-----------------------------------------------
Dealer's Hand: 9 + ?
9_Spades    ?
Your Hand: 9
3_Clubs     6_Clubs
-----------------------------------------------
Would you like to hit (H) or stand (S)?
h
-----------------------------------------------
Dealer's Hand: 9 + ?
9_Spades    ?
Your Hand: 11
3_Clubs     6_Clubs     2_Diamonds
-----------------------------------------------
Hit or Stand?
h
-----------------------------------------------
Dealer's Hand: 9 + ?
9_Spades    ?
Your Hand: 18
3_Clubs     6_Clubs     2_Diamonds  7_Clubs
-----------------------------------------------
Hit or Stand?
s
-----------------------------------------------
Dealer's Hand: 14
9_Spades    5_Hearts
Your Hand: 18
3_Clubs     6_Clubs     2_Diamonds  7_Clubs
-----------------------------------------------
Dealer draws one card.
-----------------------------------------------
Dealer's Hand: 24
9_Spades    5_Hearts    K_Hearts
Your Hand: 18
3_Clubs     6_Clubs     2_Diamonds  7_Clubs
-----------------------------------------------

Final Results!
-----------------------------------------------
Dealer's Hand: 24
5_Hearts    9_Spades    K_Hearts
Your Hand: 18
2_Diamonds  3_Clubs     6_Clubs     7_Clubs
-----------------------------------------------
Dealer busts! You win!
You receive $50.00
***********************************************************
Enter 'Y' or 'y' to play again, all other inputs quit game.
n
***********************************************************
You did not win more games than you lost.
You won 1 games out of 2
Your win percentage is 50.00%
You bet a total of $100.00
You received a total of $50.00
***********************************************

RUN SUCCESSFUL (total time: 57s)
```

Program

```
/*
 * File:   main.cpp
 * Author: Brandon Smith
 * Created on July 30
 * Purpose: Blackjack v6
 */

//System Libraries
#include <iostream>     //Input/Output library
#include <iomanip>      //Format Library
#include <cstdlib>      //Srand
#include <fstream>      //File operator
#include <string>       //String Library
#include <ctime>        //Time to set random number seed
#include <vector>       //Includes vectors

using namespace std;

//User Libraries
//Global Constants
//Mathematical/Physics/Conversions, Higher dimensioned arrays

//Function Prototypes
void ini(float&,float&,float&);                    //Function to initialize variables
void iniL(int&,int&,bool&,bool&,bool&,int&,int&); //Function to initialize variables in loop
void stars(int=47);                               //Function to display stars
void shuffle(int[],int[],const int,const int,const int);                              //Function to shuffle the
cards
void draw(int[],int[],string[],string[],const int, const int, const int);             //Function to assign
cards
void points(int[],int[],int[],int[],const int, const int);                            //Function to calculate card
point values
void user(int[],int[],string[],string[],int&,int,int&,bool&,const int,int[]);         //Function for users turn
void dealer(int[],int[],string[],string[],int&,int&,int&,int&,bool&,const int,int[]); //Function for dealers
turn
void results(int,int,bool,bool,float&,float&,vector<float>,float&);                   //Function to display
results
void sortD(int,int[],string[]);                   //Function to sort the dealers cards
void sortU(int,int[],string[]);                   //Function to sort the users cards
void printF(int,int,string[],string[],int,int);   //Function to print the final cards
float winP(float,float);                          //Function to calculate win percentage
bool mWL(float,float);                            //Function to calculate if more wins/losses
void bet(int,vector<float>,float,float&);         //Function to calculate betting
void gamble(float,vector<float>,float);          //Function for gambling results

//Execution Begins Here
int main(int argc, char** argv) {
   //Initialize the Random Number Seed
   srand(static_cast<unsigned>(time(0)));

   //Declare Variables
   const int DECK_OF_CARDS=52;            //Cards in a deck
   const int USER_CARDS=21;               //How many cards can be drawn by user
   const int DEALER_CARDS=17;             //How many cards can be drawn by dealer
   int uCardV[USER_CARDS],                //User card value array
      dCardV[DEALER_CARDS],               //Dealer card value array
      uC[USER_CARDS],                     //User card points array
      dC[DEALER_CARDS];                   //Dealer card points array
   string uCard[USER_CARDS],              //User card name array
        dCard[DEALER_CARDS];              //Dealer card name array
   int userT, dealT,                      //User total, dealer total
         cdU, cdD;                        //User cards drawn, dealer cards drawn
   float gWon, gPlay,                     //Games won, game played
        sum;                              //Sum of money received
   char again;                            //Play again
   bool uBust, dBust, win;                //User bust, dealer bust, game result
   vector<float> bets(100);

   //Game play
   ini(gWon,gPlay,sum);          //Calls initialize variables loop
   stars();                      //Calls print stars function
   do{
      iniL(userT,dealT,uBust,dBust,win,cdU,cdD);   //Calls initialize variables in loop function
      //Betting
      cout<<"How much would you like to bet? Max bet is $100."<<endl; //Input prompt
      cin>>bets[gPlay];                         //User input
      while (bets[gPlay]<0 or bets[gPlay]>100)     //Input validation
      {
         cout<<"Invalid amount, enter an amount from $0.01 to $100.00."<<endl;
         cin>>bets[gPlay];
      }
      //Shuffle and assign cards
      shuffle(uCardV,dCardV,USER_CARDS,DEALER_CARDS,DECK_OF_CARDS);
//Calls shuffle function

draw(uCardV,dCardV,uCard,dCard,USER_CARDS,DEALER_CARDS,DECK_OF_CARDS);
//Calls draw function
```

```
    points(uCardV,dCardV,uC,dC,USER_CARDS,DEALER_CARDS);            //Calls point
function
    //Initial Game output
    userT=uC[1]+uC[2];     //Initial user total
    dealT=dC[1]+dC[2];     //Initial dealer total
    //Output initial hand
    if (userT>21)          //Sets new ace value if bust
    {
      for (int uCount=0; uCount<USER_CARDS; uCount++)     //Loop to assign point values
to user cards
      {
        if (uCardV[uCount]==1  or uCardV[uCount]==14  or uCardV[uCount]==27  or
uCardV[uCount]==40)uC[uCount]=1;
      }
      userT=uC[1]+uC[2];
    }
    cout<<"--------------------------------------------"<<endl;
    cout<<"Dealer's Hand: "<<dC[1]<<" + ?"<<endl;
    cout<<left<<setw(12)<<dCard[1]<<left<<setw(12)<<"?"<<endl;
    cout<<"Your Hand: "<<userT<<endl;
    cout<<left<<setw(12)<<uCard[1]<<left<<setw(12)<<uCard[2]<<endl;
    cout<<"--------------------------------------------"<<endl;
    //Turns and results
    user(uC,dC,uCard,dCard,userT,dealT,cdU,uBust,USER_CARDS,uCardV);      //Calls
user turn function
    dealer(uC,dC,uCard,dCard,userT,dealT,cdU,cdD,dBust,USER_CARDS,dCardV);   //Calls
dealer turn function
    sortD(cdD,dC,dCard);                                    //Calls sort dealer hand function
    sortU(cdU,uC,uCard);                                    //Calls sort user hand function
    printF(dealT,userT,dCard,uCard,cdD,cdU);
    results(userT,dealT,uBust,dBust,gPlay,gWon,bets,sum);           //Calls results
function
    //Repeat play
    stars(59);
    cout<<"Enter 'Y' or 'y' to play again, "     //Play again prompt
      <<"all other inputs quit game."<<endl;
    cin>>again;                          //Play again input
    stars(59);                           //Calls print stars function
  }while (again=='Y' or again=='y');          //Loops if yes

  //Final results output
  if (mWL(gWon,gPlay)){cout<<"You won more games than you lost."<<endl;}
  else{cout<<"You did not win more games than you lost."<<endl;}
```

```
    cout<<"You won "<<static_cast<int>(gWon)                //Outputs games won
        <<" games out of "<<static_cast<int>(gPlay)<<endl;  //Outputs games played
    cout<<"Your win percentage is "                         //Outputs win%
        <<fixed<<setprecision(2)<<winP(gWon,gPlay)<<"%"<<endl;
    gamble(gPlay,bets,sum);                                 //Calls gamble results function
    stars();                                        //Calls print starts function
    //Exit stage right
    return 0;
}

//Functions
void ini(float& gWon,float& gPlay,float& sum)
{
    gWon=0;                     //Default games won
    gPlay=0;                    //Default games played
    sum=0;                      //Default sum value
}

void iniL(int& userT,int& dealT,bool& uBust,bool& dBust,bool& win,int& cdU,int& cdD)
{
    userT=0;                    //Default user total
    dealT=0;                    //Default dealer total
    uBust=false;                //Default user bust
    dBust=false;                //Default dealer bust
    win=false;                  //Default game result
    cdU=2;                      //Default user cards drawn
    cdD=2;                      //Default dealer cards drawn
}

void stars(int row)
{
    for (int i=0; i<row; i++)   //Output loop
    {
        cout<<"*";              //Outputs star
    }
    cout<<endl;                 //Formatting
}

void shuffle(int uCardV[],int dCardV[],const int USER_CARDS,const int DEALER_CARDS,const
int DECK_OF_CARDS)
{
    static int shuf;
    for (int uCount=0; uCount<USER_CARDS; uCount++)         //User card shuffle
    {
        uCardV[uCount]=(rand()%DECK_OF_CARDS+1);           //Calculates random value
    }

    for (int dCount=0; dCount<DEALER_CARDS; dCount++)      //Dealer card shuffle
    {
        dCardV[dCount]=(rand()%DECK_OF_CARDS+1);           //Calculates random value
    }
    shuf++;
    cout<<endl<<"The cards have been shuffled a total of "<<shuf<<" times."<<endl;
}

void draw(int uCardV[],int dCardV[],string uCard[],string dCard[],const int USER_CARDS,const
int DEALER_CARDS,const int DECK_OF_CARDS)
{
    //Variables for File
    fstream input;                        //File input
    string cardIn;                        //Card string
    string fileName;                      //File name
    //Initialize file parameters
    fileName="deckOfCards.dat";           //File name
    input.open(fileName.c_str(),ios::in);    //Opens file

    for (int count=1; count<=DECK_OF_CARDS; count++)        //Loop to input cards
    {
        input>>cardIn;                                  //Inputs current card
        for (int uCount=0; uCount<USER_CARDS; uCount++)     //User Cards loop
        {
            if (count==uCardV[uCount])
            {
                uCard[uCount]=cardIn; //Assigns cards
            }
        }
        for (int dCount=0; dCount<DEALER_CARDS; dCount++)   //Dealer cards loop
        {
            if (count==dCardV[dCount])
            {
                dCard[dCount]=cardIn; //Assigns cards
            }
        }
    }
    //Close the file
    input.close();
}
```

```
void points(int uCardV[],int dCardV[],int uC[],int dC[],const int USER_CARDS,const int
DEALER_CARDS)
{
    for (int uCount=0; uCount<USER_CARDS; uCount++)        //Loop to assign point values to
user cards
    {
        if (uCardV[uCount]==1  or uCardV[uCount]==14  or uCardV[uCount]==27  or
uCardV[uCount]==40)uC[uCount]=11;
        else if (uCardV[uCount]==2  or uCardV[uCount]==15  or uCardV[uCount]==28  or
uCardV[uCount]==41)uC[uCount]=2;
        else if (uCardV[uCount]==3  or uCardV[uCount]==16  or uCardV[uCount]==29  or
uCardV[uCount]==42)uC[uCount]=3;
        else if (uCardV[uCount]==4  or uCardV[uCount]==17  or uCardV[uCount]==30  or
uCardV[uCount]==43)uC[uCount]=4;
        else if (uCardV[uCount]==5  or uCardV[uCount]==18  or uCardV[uCount]==31  or
uCardV[uCount]==44)uC[uCount]=5;
        else if (uCardV[uCount]==6  or uCardV[uCount]==19  or uCardV[uCount]==32  or
uCardV[uCount]==45)uC[uCount]=6;
        else if (uCardV[uCount]==7  or uCardV[uCount]==20  or uCardV[uCount]==33  or
uCardV[uCount]==46)uC[uCount]=7;
        else if (uCardV[uCount]==8  or uCardV[uCount]==21  or uCardV[uCount]==34  or
uCardV[uCount]==47)uC[uCount]=8;
        else if (uCardV[uCount]==9  or uCardV[uCount]==22  or uCardV[uCount]==35  or
uCardV[uCount]==48)uC[uCount]=9;
        else uC[uCount]=10;
    }
    for (int dCount=0; dCount<DEALER_CARDS; dCount++)        //Loop to assign point values
to user cards
    {
        if (dCardV[dCount]==1  or dCardV[dCount]==14  or dCardV[dCount]==27  or
dCardV[dCount]==40)dC[dCount]=11;
        else if (dCardV[dCount]==2  or dCardV[dCount]==15  or dCardV[dCount]==28  or
dCardV[dCount]==41)dC[dCount]=2;
        else if (dCardV[dCount]==3  or dCardV[dCount]==16  or dCardV[dCount]==29  or
dCardV[dCount]==42)dC[dCount]=3;
        else if (dCardV[dCount]==4  or dCardV[dCount]==17  or dCardV[dCount]==30  or
dCardV[dCount]==43)dC[dCount]=4;
        else if (dCardV[dCount]==5  or dCardV[dCount]==18  or dCardV[dCount]==31  or
dCardV[dCount]==44)dC[dCount]=5;
        else if (dCardV[dCount]==6  or dCardV[dCount]==19  or dCardV[dCount]==32  or
dCardV[dCount]==45)dC[dCount]=6;
        else if (dCardV[dCount]==7  or dCardV[dCount]==20  or dCardV[dCount]==33  or
dCardV[dCount]==46)dC[dCount]=7;
```

```
      else if (dCardV[dCount]==8  or dCardV[dCount]==21  or dCardV[dCount]==34  or
dCardV[dCount]==47)dC[dCount]=8;
      else if (dCardV[dCount]==9  or dCardV[dCount]==22  or dCardV[dCount]==35  or
dCardV[dCount]==48)dC[dCount]=9;
      else dC[dCount]=10;
   }
}

void user(int uC[],int dC[],string uCard[],string dCard[],int& userT,int dealT,int& cdU,bool&
uBust,const int USER_CARDS,int uCardV[])
{
   char hOs;               //Initial hit or stand
   char repeat;            //Repeat Hit or Stand
   cdU=2;                  //Default cards drawn by user
   cout<<"Would you like to hit (H) or stand (S)?"<<endl;      //Input prompt
   cin>>hOs;                                                   //Input
   while (hOs!='H' && hOs!='h' && hOs!='S' && hOs!='s')        //Input validation
   {
      cout<<"Invalid choice, please enter 'H' for hit or 'S' for stand."<<endl;
      cin>>hOs;
   }
   switch (hOs)                       //Hit or Stand switch
   {
      case 'H':                       //Case Hit
      case 'h':
         do{
            cdU++;                    //Cards drawn by user plus one
            userT+=uC[cdU];           //Adds new card to total
            if (userT>21)             //Sets new ace value if bust
            {
               for (int uCount=0; uCount<USER_CARDS; uCount++)         //Loop to assign point
values to user cards
               {
                  if (uCardV[uCount]==1  or uCardV[uCount]==14  or uCardV[uCount]==27  or
uCardV[uCount]==40)uC[uCount]=1;
               }
               userT=0;                    //Reset user total
               for (int i=1; i<=cdU; i++)
               {
                  userT+=uC[i];
               }
            }
            //Outputs new hand
            cout<<"--------------------------------------------"<<endl;
```

```
        cout<<"Dealer's Hand: "<<dC[1]<<" + ?"<<endl;
        cout<<left<<setw(12)<<dCard[1]<<left<<setw(12)<<"?"<<endl;
        cout<<"Your Hand: "<<userT<<endl;
        cout<<left<<setw(12)<<uCard[1]<<left<<setw(12)<<uCard[2];
        for (int i=3; i<=cdU; i++)                    //Loop to display user cards
        {
            if (i==5 or i==9 or i==13 or i==17 or i==21)    //Spacing
            {cout<<endl;}
            cout<<left<<setw(12)<<uCard[i];

        }
        cout<<endl;
        cout<<"----------------------------------------------"<<endl;
        if (userT<=21)uBust=false;     //If user does not bust, sets bool to false
        if (userT>21)
        {
            uBust=true;               //If user busts, sets bool to true
            exit;
        }
        if (userT<21)                 //Only allows choice if user is under 21
        {
            cout<<"Hit or Stand?"<<endl;
            cin>>repeat;
            while (repeat!='H' && repeat!='h' && repeat!='S' && repeat!='s')     //Input validation
            {
                cout<<"Invalid choice, please enter 'H' for hit or 'S' for stand."<<endl;
                cin>>repeat;
            }
        }
        else{repeat='S';}             //Forces stand if already busted
    }while (repeat=='H' or repeat=='h');    //Repeats if user hits again

case 'S':                   //Case Stand
case 's':
    //Output initial hand and reveal dealers hand
    cout<<"----------------------------------------------"<<endl;
    cout<<"Dealer's Hand: "<<dealT<<endl;
    cout<<left<<setw(12)<<dCard[1]<<left<<setw(12)<<dCard[2]<<endl;
    cout<<"Your Hand: "<<userT<<endl;
    cout<<left<<setw(12)<<uCard[1]<<left<<setw(12)<<uCard[2];
    for (int i=3; i<=cdU; i++)      //Outputs additional cards drawn by user
        {
            if (i==5 or i==9 or i==13 or i==17 or i==21)    //Spacing
            {cout<<endl;}
```

```
          cout<<left<<setw(12)<<uCard[i];
        }
      cout<<endl;
      cout<<"---------------------------------------------"<<endl;
  }
}

void dealer(int uC[],int dC[],string uCard[],string dCard[],int& userT,int& dealT,int& cdU,int&
cdD,bool& dBust,const int USER_CARDS,int dCardV[])
{
  cdD=2;                    //Default cards drawn by dealer
  if (dealT<17 && userT<22)   //If dealer is under 17 and user did not bust
  {
    do{
    cdD++;                  //Adds one to dealer cards drawn
    dealT+=dC[cdD];         //Adds new card to dealer total
    cout<<"Dealer draws one card."<<endl;   //Dealer draw prompt
    if (dealT>21)           //Sets new ace value if bust
        {
          for (int dCount=0; dCount<USER_CARDS; dCount++)         //Loop to assign point
values to dealer cards
          {
            if (dCardV[dCount]==1  or dCardV[dCount]==14  or dCardV[dCount]==27  or
dCardV[dCount]==40)dC[dCount]=1;
          }
          dealT=0;                  //Reset user total
          for (int i=1; i<=cdD; i++)
          {
            dealT+=dC[i];
          }
        }
    //Outputs new hand
    cout<<"---------------------------------------------"<<endl;
    cout<<"Dealer's Hand: "<<dealT<<endl;
    cout<<left<<setw(12)<<dCard[1]<<left<<setw(12)<<dCard[2];
    for (int i=3; i<=cdD; i++)          //Loop to output additional cards
    {
      if (i==5 or i==9 or i==13 or i==17 or i==21)    //Spacing
      {cout<<endl;}
      cout<<left<<setw(12)<<dCard[i];
    }
    cout<<endl;
    cout<<"Your Hand: "<<userT<<endl;
    cout<<left<<setw(12)<<uCard[1]<<left<<setw(12)<<uCard[2];
    for (int i=3; i<=cdU; i++)          //Loop to output additional cards
      {
        if (i==5 or i==9 or i==13 or i==17 or i==21)//Spacing
        {cout<<endl;}
        cout<<left<<setw(12)<<uCard[i];
      }
    cout<<endl;
    cout<<"--------------------------------------------"<<endl;
    }while (dealT<17);      //Repeats if dealer is still under 17
  }
  if (dealT<=21)dBust=false;  //If dealer does not bust, sets bool to false
  if (dealT>21)dBust=true;    //If dealer busts, sets bool to true
}

void sortD(int cdD,int dC[],string dCard[])
{
  //Bubble Sort
  bool swap;          //Swap bool
  int temp;           //Temporary placeholder
  string temp2;       //Temporary placeholder

  do{                 //Bubble sort loop
    swap=false;      //Default swap value
    for (int count=1; count<cdD; count++)    //Loop to check values
    {
      if (dC[count]>dC[count+1])           //If greater than next array
      {
        temp=dC[count];                  //Holds first array value temporarily
        temp2=dCard[count];              //Holds first array value temporarily

        dC[count]=dC[count+1];           //Sets first array equal to second
        dCard[count]=dCard[count+1];     //Sets first array equal to second

        dC[count+1]=temp;                //Sets second array equal to original first
        dCard[count+1]=temp2;            //Sets second array equal to original first
        swap=true;                       //Sets swap to true
      }
    }
  }while (swap);
}

void sortU(int cdU,int uC[],string uCard[])
{
  //Selection Sort
```

```
    int sScan, minI, minV;              //Start scan, min index, min value
    string temp;                        //Temporary placeholder
    for (sScan=1; sScan<cdU; sScan++)   //Selection Sort loop
    {
        minI=sScan;         //Min index default value
        minV=uC[sScan];     //Min value default value
        temp=uCard[sScan];  //Placeholder default value
        for (int index=sScan+1; index<=cdU; index++)  //Loop to check values
        {
            if (uC[index]<minV)      //If next value is less than previous
            {
                minV=uC[index];         //Minimum value set to array value
                minI=index;             //Minimum index set to current index
                temp=uCard[index];      //String value set to array value
            }
        }
        uC[minI]=uC[sScan];             //Sets new values
        uCard[minI]=uCard[sScan];       //Sets new values
        uC[sScan]=minV;                 //Sets new values
        uCard[sScan]=temp;              //Sets new values
    }
}

void printF(int dealT,int userT,string dCard[],string uCard[],int cdD,int cdU)
{
    cout<<endl<<"Final Results!"<<endl;
    cout<<"--------------------------------------------------"<<endl;
    cout<<"Dealer's Hand: "<<dealT<<endl;
    cout<<left<<setw(12)<<dCard[1]<<left<<setw(12)<<dCard[2];
    for (int i=3; i<=cdD; i++)          //Loop to output additional cards
    {
        if (i==5 or i==9 or i==13 or i==17 or i==21)    //Spacing
        {cout<<endl;}
        cout<<left<<setw(12)<<dCard[i];
    }
    cout<<endl;
    cout<<"Your Hand: "<<userT<<endl;
    cout<<left<<setw(12)<<uCard[1]<<left<<setw(12)<<uCard[2];
    for (int i=3; i<=cdU; i++)          //Loop to output additional cards
        {
            if (i==5 or i==9 or i==13 or i==17 or i==21)//Spacing
            {cout<<endl;}
            cout<<left<<setw(12)<<uCard[i];
        }
```

```
   cout<<endl;
   cout<<"---------------------------------------------"<<endl;
}

void results(int userT,int dealT,bool uBust,bool dBust,float& gPlay,float& gWon,vector<float>
bets,float& sum)
{

   string winH="You win, congratulations!",            //Win by higher number message
        winB="Dealer busts! You win!",               //Win by dealer bust message
        lossH="You lost, better luck next time",     //Loss by lower number message
        lossB="You bust! Better luck next time",     //Loss by user bust message
        draw="It's a draw!";                         //Draw message
   if (uBust!=true)                  //If user does not  busts
   {
      if (dBust==true)               //If dealer busts
      {
         cout<<winB<<endl;           //Output message
         gWon++;                    //Adds 1 to games won counter
         bet(2,bets,gPlay,sum);      //Bet results
      }
      if (dBust==false)              //If dealer does not bust
      {
         if (userT>dealT)            //If user total higher than dealer
         {
            cout<<winH<<endl;       //Output message
            gWon++;                 //Adds 1 to games won counter
            bet(2,bets,gPlay,sum);  //Bet results
         }
         else if (userT<dealT)       //If user total less than dealer
         {
            cout<<lossH<<endl;      //Output message
            bet(0,bets,gPlay,sum);  //Bet results
         }
         else if (userT==dealT)      //If user total equal to dealer
         {
            cout<<draw<<endl;       //Output message
            bet(1,bets,gPlay,sum);  //Bet results
         }
      }
   }
   else                              //If user busts
   {
      cout<<lossB<<endl;
      bet(0,bets,gPlay,sum);          //Bet results
   }
   gPlay++;                           //Adds 1 to games played counter
}

void bet(int num, vector<float> bets,float gPlay,float& sum)
{
   cout<<"You receive $"<<fixed<<setprecision(2)<<(bets[gPlay]*num)<<endl; //Current bet
results
   sum+=bets[gPlay]*num;           //Adds result to total sum
}

void gamble(float gPlay,vector<float> bets,float sum)
{
   float total=0;                         //Initial total value
   for (int count=0; count<gPlay; count++) //Loop for total bet value
   {
      total+=bets[count];
   }
   cout<<"You bet a total of $"<<fixed<<setprecision(2)<<total<<endl;      //Total bets output
   cout<<"You received a total of $"<<fixed<<setprecision(2)<<sum<<endl;   //Total money
received output
}

bool mWL(float gWon,float gPlay)
{
   bool status;                   //Win loss status
   int gLoss=gPlay-gWon;           //Games loss calculation
   if (gWon<gLoss)status=false;    //If more games lost, return false
   if (gWon>gLoss)status=true;     //If more games won, return true
   return status;                  //Return status
}

float winP(float gWon,float gPlay)
{
   float percent;                 //Percentage
   percent=(gWon/gPlay)*100;       //Percentage calculation
   return percent;                 //Returns percentage
}
```