Project 1

Title

Blackjack

Course

CIS-5

Section

45428

Due Date

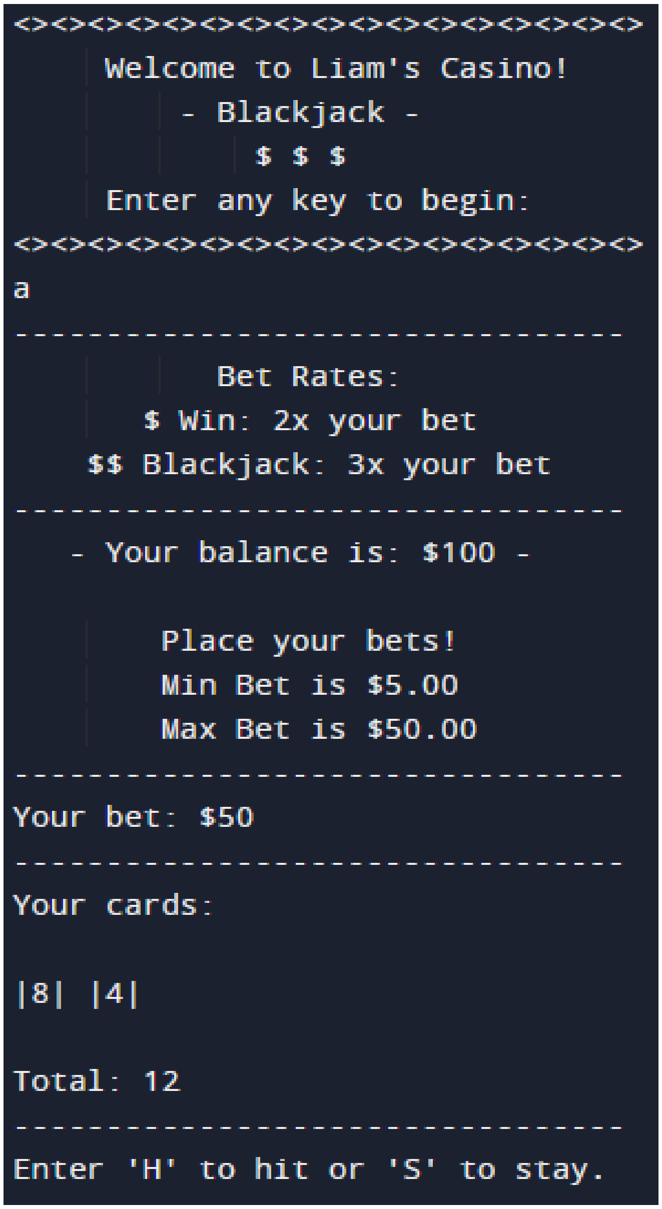
July 23, 2023

Author

Liam Shaw

1 Introduction

Blackjack, also called twenty one is a popular card game that people enjoy playing in both casinos and their own homes all over the world. It's a game that combines simplicity and excitement. The goal is to outsmart the dealer by getting a hand value as to 21 as possible without going over.



```
<><><><><><><><><><><><><><><><><><>
     Welcome to Liam's Casino!
         - Blackjack -
             $ $ $
     Enter any key to begin:
<><><><><><><><><><><><><><><><><><>
a
----------------------------------
           Bet Rates:
       $ Win: 2x your bet
    $$ Blackjack: 3x your bet
----------------------------------
   - Your balance is: $100 -

        Place your bets!
        Min Bet is $5.00
        Max Bet is $50.00
----------------------------------
Your bet: $50
----------------------------------
Your cards:

|8| |4|

Total: 12
----------------------------------
Enter 'H' to hit or 'S' to stay.
```

2 Gameplay and rules

At the start of the game the player has the option to place a bet between $5 and $50. The payout is 2x for a regular against the dealer, and 3x for a blackjack (aka, exactly 21). After confirming the bet, the player receives two random cards. They have the option to request more cards one at a time until they decide to stop or until their hand exceeds 21 resulting in an automatic loss referred to as a "bust."

In blackjack each card has a value equivalent to its face value except for face cards (Kings, Queens and Jacks) which are worth 10 points. The Ace can be counted as either 1 or 11 depending on how it suits the players strategy. The dealer follows rules when drawing cards and players must make strategic decisions on when to hit (request another card) stand (stop requesting more cards). This paced game with elements of skill makes it an exciting choice, for experienced gamblers and casual players alike.

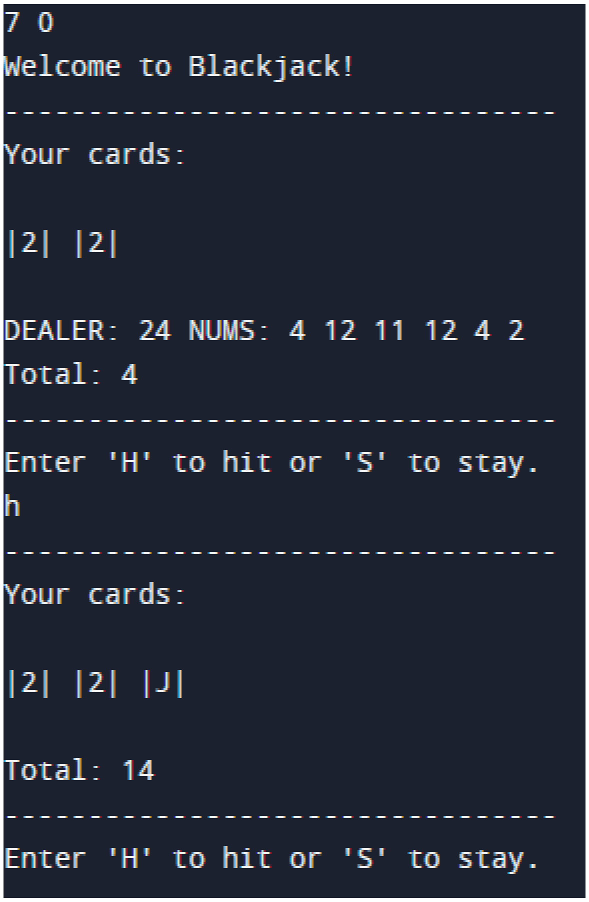
3 Development Summary

Lines of Code	**269** - (69.33%)
Comment Lines	**81 -** (20.88%)
Blank Lines	**38** - (9.79%)
Total Lines	**388**

Creating this program marked my first venture into C++ programming and game development. While it did take a considerable amount of time, the end result turned out to be simpler than anticipated. It occupied approximately 15 hours of effort to complete and Utilizing the Netbeans IDE, I relied on several libraries, including <iostream>, <iomanip>, <cmath>, <cstdlib>, <fstream>, <string>, and <ctime>.

Thankfully, my previous experience with Python facilitated the logical aspects of the project, making it relatively easier to grasp. However, I encountered challenges with formatting, which slowed down my progress and affected my overall performance. Despite these obstacles, I persevered and successfully completed the program, gaining valuable experience in the process.

V1.1: Barebones Program



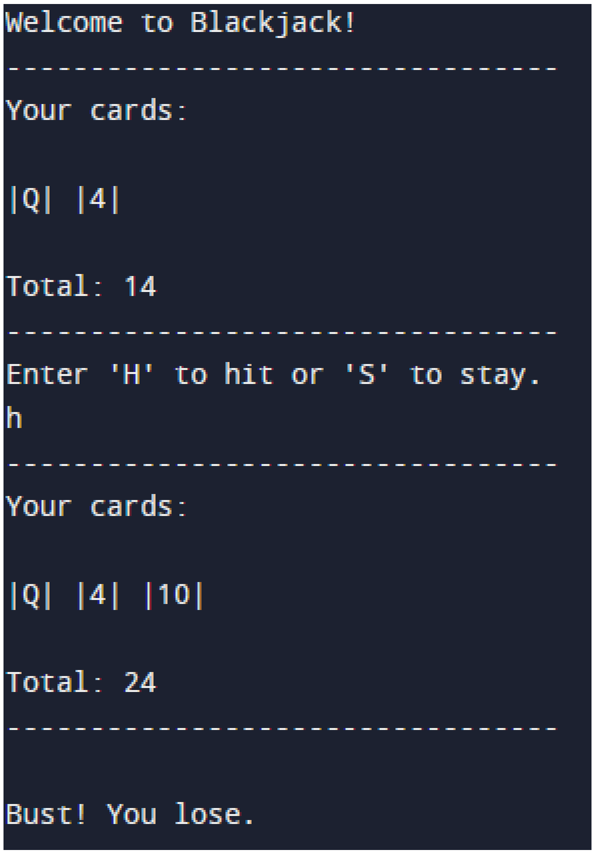
```
7 0
Welcome to Blackjack!
---------------------------------
Your cards:

|2| |2|

DEALER: 24 NUMS: 4 12 11 12 4 2
Total: 4
---------------------------------
Enter 'H' to hit or 'S' to stay.
h
---------------------------------
Your cards:

|2| |2| |J|

Total: 14
---------------------------------
Enter 'H' to hit or 'S' to stay.
```

The program was initiated by establishing the basic structure. The first step involved developing the random number generator, followed by creating the start menu. Next, a list was implemented to store the necessary values. Then, a mechanism to compute corresponding values was incorporated. Subsequently, various 'if' statements were employed to determine successful hits when required.

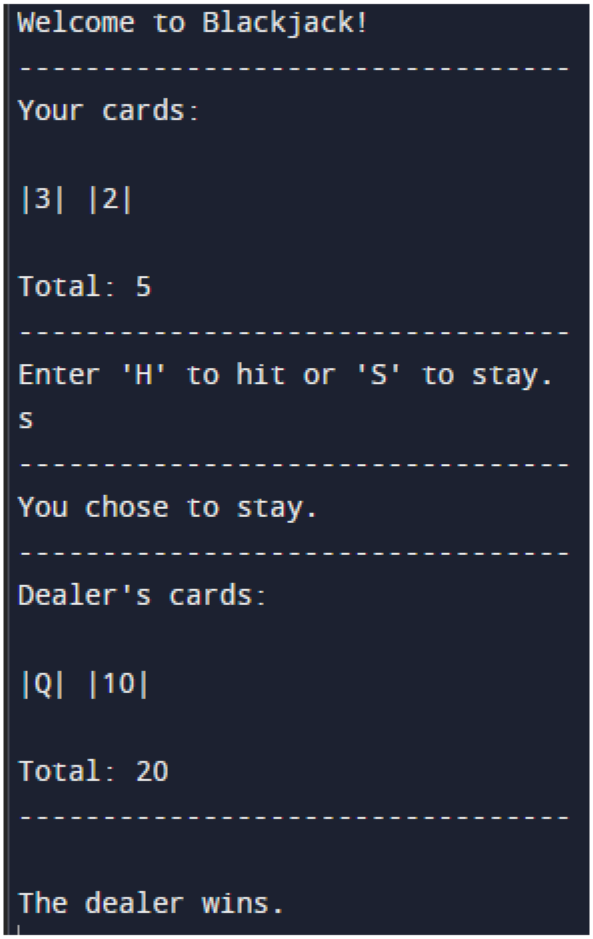
As progress continued, attention shifted towards enabling the dealer to play. However, during the process, it was realized that using 'for' loops would be a more efficient approach to save time. Consequently, a decision was made to implement 'for' loops, resulting in the creation of a new version.

An important aspect to mention is that during the development phase, I forced the initial cards to be set at 2 to test the hit or stay system. As part of this process, I also displayed the dealer's card values to ensure the random values were functioning correctly for the dealer as well. Consequently, the output during testing may be confusing and seemingly inconsistent, but this approach was deliberately maintained to focus on pursuing version 1.2 and refining the program further.

V1.2: For Loops, Functions, Dealers, and Hit/Stay



```
Welcome to Blackjack!
---------------------------------
Your cards:

|Q| |4|

Total: 14
---------------------------------
Enter 'H' to hit or 'S' to stay.
h
---------------------------------
Your cards:

|Q| |4| |10|

Total: 24
---------------------------------


Bust! You lose.
```



```
Welcome to Blackjack!
---------------------------------
Your cards:

|3| |2|

Total: 5
---------------------------------
Enter 'H' to hit or 'S' to stay.
s
---------------------------------
You chose to stay.
---------------------------------
Dealer's cards:

|Q| |10|

Total: 20
---------------------------------


The dealer wins.
```

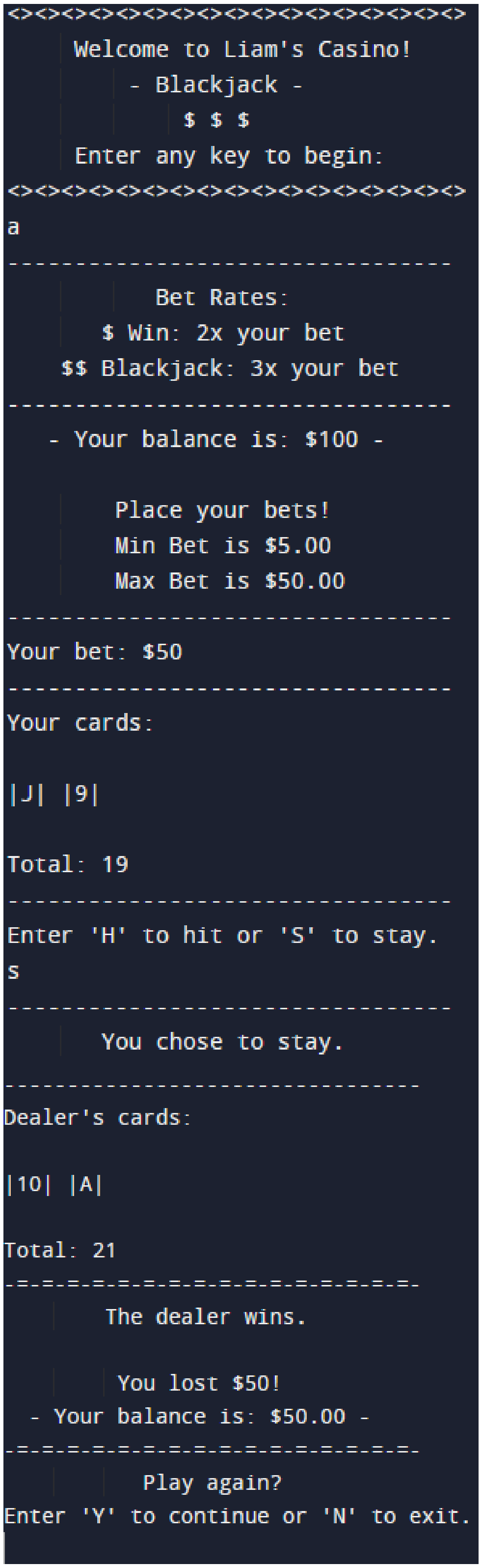
In version 1.2, significant improvements were made to enhance the program's efficiency and functionality. The first major enhancement was the implementation of external functions for random card generation. This decision was made to avoid the creation of multiple random numbers unnecessarily. Additionally, a function was introduced to calculate the total value of the cards.

Also, the variables underwent a substantial transformation, now utilizing arrays and becoming compatible with 'for' loops. This adjustment aimed at streamlining the code and improving its readability. As part of the user interface, the option to "stay" was incorporated. An 'else' statement was also added to handle any input other than 'h' (hit) or 's' (stay).

To further enhance the gaming experience, the program now displays the dealer's cards, allowing players to compare their hand against the dealer's and determine the outcome (win, lose, or tie). However, I decided not to reveal the dealer's cards if the player busts or achieves a blackjack, as such an action would be unnecessary.

These significant updates have improved the program's efficiency, user experience, and overall performance.

V1.3: UI and Gambling



```
<><><><><><><><><><><><><><><><><><><><><><><><>
     Welcome to Liam's Casino!
         - Blackjack -
             $ $ $
     Enter any key to begin:
<><><><><><><><><><><><><><><><><><><><><><><><>
a
---------------------------------
          Bet Rates:
      $ Win: 2x your bet
   $$ Blackjack: 3x your bet
---------------------------------
   - Your balance is: $100 -

        Place your bets!
        Min Bet is $5.00
        Max Bet is $50.00
---------------------------------
Your bet: $50
---------------------------------
Your cards:

|J| |9|

Total: 19
---------------------------------
Enter 'H' to hit or 'S' to stay.
s
---------------------------------
       You chose to stay.
-------------------------------
Dealer's cards:

|10| |A|

Total: 21
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
       The dealer wins.

        You lost $50!
  - Your balance is: $50.00 -
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
          Play again?
Enter 'Y' to continue or 'N' to exit.
```

Version 1.3 of the program focused on enhancing the user interface and introducing a betting system for a more engaging experience. Initially, several lines of strings were added to provide additional information and personalize the UI. Additionally, new lines ('\n') were removed to create a more visually appealing layout. The presentation was further improved by centering results like "Win" and "Lose" with proper spacing.

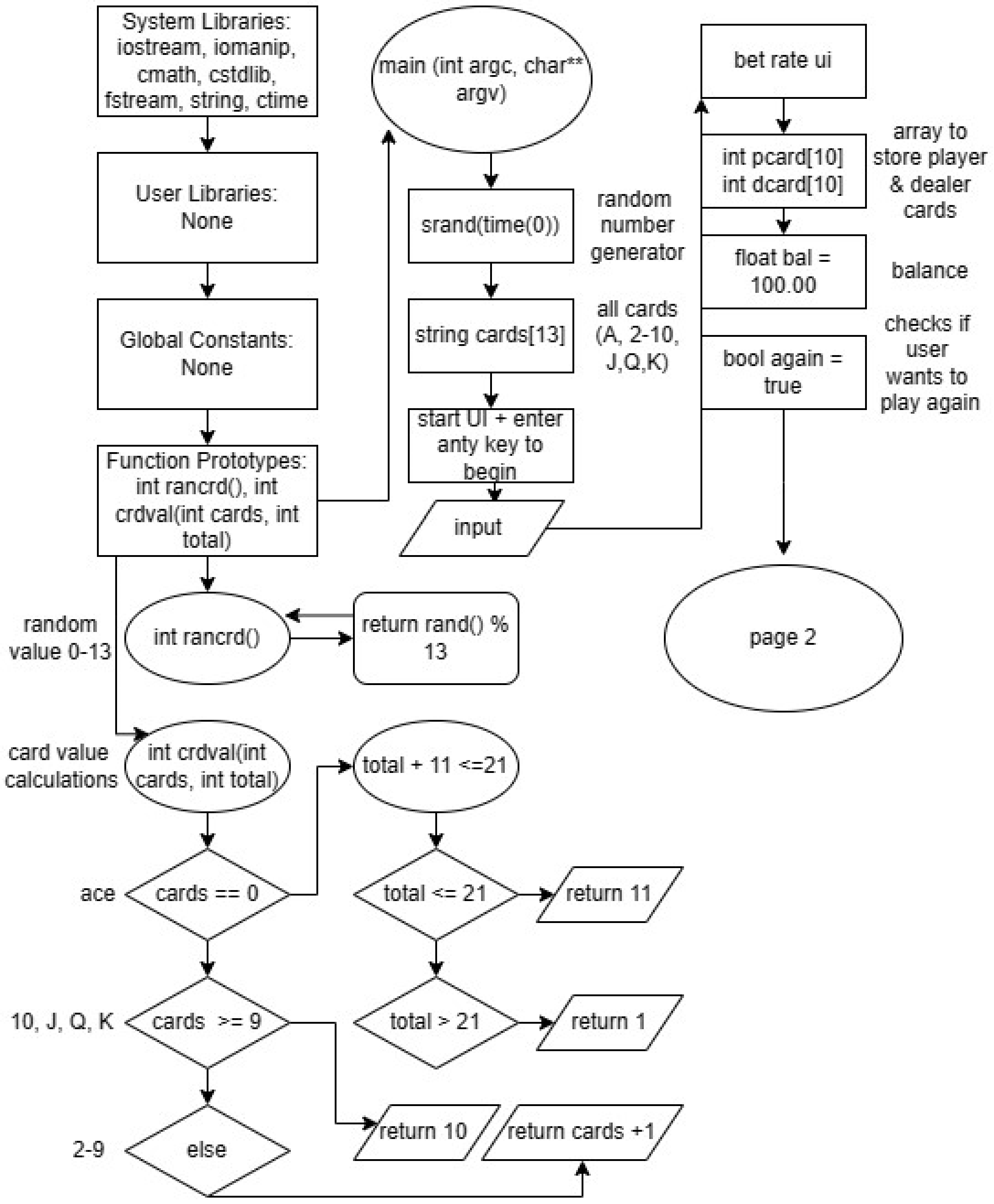
The main highlight of this version was the implementation of a functional betting system. A simple balance system was established to keep track of the player's money. The UI was enhanced to display and handle bets effectively. Players could now continue playing after each round, and the program ensured that invalid choices were prevented. Furthermore, a loop was introduced to end the game if the player's available funds were insufficient for placing a bet.

To provide clarity to players, a starting UI was added to indicate the bet rates—2x on a win and 3x on a blackjack. These additions transformed the game into a more immersive and enjoyable gaming experience with improved aesthetics and a fully functional betting system.

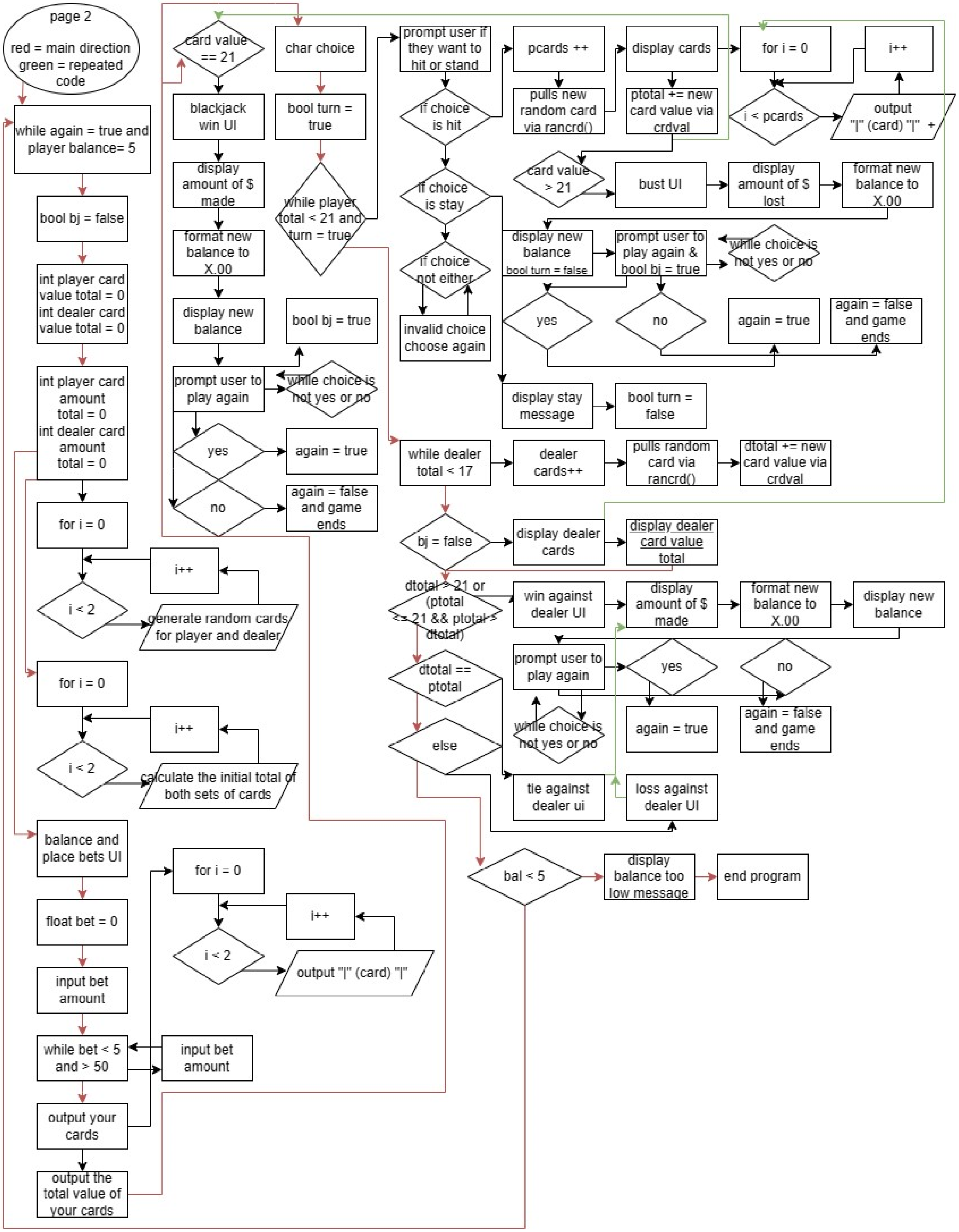
Flowchart Page 1

Author: Liam Shaw
Created on 7/22/23
Purpose: A functional and fun to use C++ Blackjack game.

Project 1: Blackjack


System Libraries: iostream, iomanip, cmath, cstdlib, fstream, string, ctime
User Libraries: None
Global Constants: None
Function Prototypes: int rancrd(), int crdval(int cards, int total)
main (int argc, char** argv)
srand(time(0))
random number generator
string cards[13]
all cards (A, 2-10, J,Q,K)
start UI + enter anty key to begin
input
bet rate ui
int pcard[10] int dcard[10]
array to store player & dealer cards
float bal = 100.00
balance
bool again = true
checks if user wants to play again
page 2
random value 0-13
int rancrd()
return rand() % 13
card value calculations
int crdval(int cards, int total)
total + 11 <=21
ace
cards == 0
total <= 21
return 11
10, J, Q, K
cards >= 9
total > 21
return 1
return 10
return cards +1
2-9
else

Flowchart Page 2


page 2
red = main direction
green = repeated code
while again = true and player balance= 5
bool bj = false
int player card value total = 0
int dealer card value total = 0
int player card amount total = 0
int dealer card amount total = 0
for i = 0
i++
i < 2
generate random cards for player and dealer
for i = 0
i++
i < 2
calculate the initial total of both sets of cards
balance and place bets UI
float bet = 0
input bet amount
while bet < 5 and > 50
input bet amount
output your cards
output the total value of your cards
for i = 0
i++
i < 2
output "|" (card) "|"
card value == 21
blackjack win UI
display amount of $ made
format new balance to X.00
display new balance
prompt user to play again
while choice is not yes or no
bool bj = true
yes
again = true
no
again = false and game ends
char choice
bool turn = true
while player total < 21 and turn = true
prompt user if they want to hit or stand
if choice is hit
if choice is stay
if choice not either
invalid choice choose again
pcards ++
pulls new random card via rancrd()
display cards
ptotal += new card value via crdval
for i = 0
i++
i < pcards
output "|" (card) "|" +
card value > 21
bust UI
display amount of $ lost
format new balance to X.00
display new balance
bool turn = false
prompt user to play again & bool bj = true
while choice is not yes or no
yes
no
again = true
again = false and game ends
display stay message
bool turn = false
while dealer total < 17
dealer cards++
pulls random card via rancrd()
dtotal += new card value via crdval
bj = false
display dealer cards
display dealer card value total
dtotal > 21 or (ptotal <= 21 && ptotal > dtotal)
win against dealer UI
display amount of $ made
format new balance to X.00
display new balance
prompt user to play again
while choice is not yes or no
yes
no
again = true
again = false and game ends
dtotal == ptotal
else
tie against dealer ui
loss against dealer UI
bal < 5
display balance too low message
end program

Pseudo Code

```
// Function to generate a random card index from 0 to 12 (representing A to K)
Function GenerateRandomCardIndex()
    Return random number from 0 to 12

// Function to get the value of a card
Function GetCardValue(card, total)
    If card is an Ace
        If (total + 11) <= 21
            Return 11
        Else
            Return 1
    Else if card is 10, Jack, Queen, or King
        Return 10
    Else
        Return (card + 1)

Main Function
    Initialize cards array with string values for the cards
    Initialize player's card array with size 10
    Initialize dealer's card array with size 10
    Initialize balance with 100.00
    Initialize again flag as true

    // Loop while the player wants to play again and balance is more than the minimum bet
    While (again is true and balance > 5)
        Initialize blackjack flag as false
        Initialize player's total as 0
        Initialize dealer's total as 0
        Initialize player's card count as 1
        Initialize dealer's card count as 1

        // Generate two random cards for the player and dealer
        For i from 0 to 1
            player's card[i] = GenerateRandomCardIndex()
            dealer's card[i] = GenerateRandomCardIndex()

        // Calculate the initial total of the player's cards
        For i from 0 to 1
            player's total += GetCardValue(player's card[i], player's total)
            dealer's total += GetCardValue(dealer's card[i], dealer's total)
```

```
// Display initial balance and accept player's bet
Display "Your balance is: $" + balance
Display "Place your bets!"
Display "Min Bet is $5.00"
Display "Max Bet is $50.00"
Read bet from the player

// Ensure the bet is within the range
While bet < 5 or bet > 50
    Display "Please choose an amount between $5.00 and $50.00"
    Read bet from the player

// Display player's cards and total
Display "Your cards:"
For i from 0 to 1
    Display card value for player's card[i]
Display "Total: " + player's total

// Check for immediate Blackjack
If player's total is 21
    Display "Lucky you, first try Blackjack!"
    Display "You win!"
    Update balance with (bet * 3)
    Display "You made $" + (bet * 3)
    Display "Your balance is: $" + balance
    Display "Play again? Enter 'Y' to continue or 'N' to exit."
    Read choice from the player
    While choice is not 'Y', 'y', 'N', or 'n'
        Display "Invalid choice, Enter 'Y' to continue or 'N' to exit."
        Read choice from the player
    If choice is 'N' or 'n'
        Set again flag to false
    Set blackjack flag to true

// Player's turn to hit or stay
Set turn flag to true
While player's total < 21 and turn is true
    Display "Enter 'H' to hit or 'S' to stay."
    Read choice from the player
    If choice is 'H' or 'h'
        Increment player's card count
        player's card[player's card count] = GenerateRandomCardIndex()
        For i from 0 to player's card count
            Display card value for player's card[i]
```

```
                Update player's total with GetCardValue(player's card[player's card count], player's total)
                Display "Total: " + player's total
                If player's total is 21
                    Display "Blackjack! You win!"
                    Update balance with (bet * 3)
                    Display "You made $" + (bet * 3)
                    Display "Your balance is: $" + balance
                    Display "Play again? Enter 'Y' to continue or 'N' to exit."
                    Read choice from the player
                    While choice is not 'Y', 'y', 'N', or 'n'
                        Display "Invalid choice, Enter 'Y' to continue or 'N' to exit."
                        Read choice from the player
                    If choice is 'N' or 'n'
                        Set again flag to false
                    Set blackjack flag to true
                    Set turn flag to false
                Else If player's total is greater than 21
                    Display "Bust! You lose."
                    Update balance with -bet
                    Display "You lost $" + bet
                    Display "Your balance is: $" + balance
                    Display "Play again? Enter 'Y' to continue or 'N' to exit."
                    Read choice from the player
                    While choice is not 'Y', 'y', 'N', or 'n'
                        Display "Invalid choice, Enter 'Y' to continue or 'N' to exit."
                        Read choice from the player
                    If choice is 'N' or 'n'
                        Set again flag to false
                    Set blackjack flag to true
                    Set turn flag to false
            Else If choice is 'S' or 's'
                Display "You chose to stay."
                Set turn flag to false
            Else
                Display "Invalid choice. Please enter 'H' to hit or 'S' to stay."

        While dealer's total < 17
            Increment dealer's card count
            dealer's card[dealer's card count] = GenerateRandomCardIndex()
            Update dealer's total with GetCardValue(dealer's card[dealer's card count], dealer's total)

        // Determine the winner
        If blackjack flag is false
            Display "Dealer's cards:"
            For i from 0 to dealer's card count
                Display card value for dealer's card[i]
            Display "Total: " + dealer's total
            If dealer's total is greater than 21 or (player's total <= 21 and player's total > dealer's total)
                Display "Congratulations! You win!"
                Update balance with (bet * 2)
                Display "You made $" + (bet * 2)
                Display "Your balance is: $" + balance
                Display "Play again? Enter 'Y' to continue or 'N' to exit."
                Read choice from the player
                While choice is not 'Y', 'y', 'N', or 'n'
                    Display "Invalid choice, Enter 'Y' to continue or 'N' to exit."
                    Read choice from the player
                If choice is 'N' or 'n'
                    Set again flag to false
            Else If player's total is equal to dealer's total
                Display "It's a tie!"
                Display "Play again? Enter 'Y' to continue or 'N' to exit."
                Read choice from the player
                While choice is not 'Y', 'y', 'N', or 'n'
                    Display "Invalid choice, Enter 'Y' to continue or 'N' to exit."
                    Read choice from the player
                If choice is 'N' or 'n'
                    Set again flag to false
            Else
                Display "The dealer wins."
                Update balance with -bet
                Display "You lost $" + bet
                Display "Your balance is: $" + balance
                Display "Play again? Enter 'Y' to continue or 'N' to exit."
                Read choice from the player
                While choice is not 'Y', 'y', 'N', or 'n'
                    Display "Invalid choice, Enter 'Y' to continue or 'N' to exit."
                    Read choice from the player
                If choice is 'N' or 'n'
                    Set again flag to false

    If balance less than 5
        Display "Sorry! Your balance is lower than the minimum bet."
End Main Function
```

Cross Reference

Chapter	Section	Topic	Where Line #"s	Pts	Notes
2	2	cout	47…382		
	3	libraries	10-16	8	iostream, iomanip, cmath, cstdlib, fstream, string, ctime
	4	variables/literals	66,64…		No variables in global area, failed project!
	5	Identifiers	66,64…		
	6	Integers	62,64…	3	
	7	Characters	139…	3	
	8	Strings	44	3	
	9	Floats No Doubles	66…	3	Using doubles will fail the project, floats OK!
	10	Bools	68…	4	
	11	Sizeof *****			
	12	Variables 7 characters or less			All variables <= 7 characters
	13	Scope ***** No Global Variables			
	14	Arithmetic operators	127,133 …		
	15	Comments 20%+		5	Model as pseudo code
	16	Named Constants			All Local, only Conversions/Physics/Math in Global area
	17	Programming Style ***** Emulate			Emulate style in book/in class repository

3	1	cin	53…		
	2	Math Expression	127,133 …		
	3	Mixing data types ****			
	4	Overflow/Underflow ****			
	5	Type Casting		4	
	6	Multiple assignment *****			
	7	Formatting output	195…	4	
	8	Strings	44	3	
	9	Math Library	12	4	All libraries included have to be used
	10	Hand tracing ******			
4	1	Relational Operators			
	2	if	29…	4	Independent if
	4	If-else	32…	4	
	5	Nesting	175…	4	
	6	If-else-if	247…	4	
	7	Flags *****			
	8	Logical operators	240…	4	
	11	Validating user input	240…	4	
	13	Conditional Operator	240…	4	
	14	Switch		4	

5	*1*	*Increment/Decrement*	*183…*	*4*	
	2	*While*	*71…*	*4*	
	5	*Do-while*		*4*	
	6	*For loop*	*85…*	*4*	
	11	*Files input/output both*		*8*	
	12	*No breaks in loops *******			*Failed Project if included*
******* *Not required to show*			*Total*	*100*	

Program

```
/*
 * File:   main.cpp
 * Author: Liam Shaw
 *
 * Created on July, 22
 * Purpose: The classic card game, Blackjack.
 */

//System Libraries
#include <iostream>
#include <iomanip>
#include <cmath>
#include <cstdlib>
#include <fstream>
#include <string>
#include <ctime>

//Function Prototypes
using namespace std;

// Function to generate a random card index from 0 to 12 (representing A to K)
int rancrd() {
    return rand() % 13;
}

// Function to get the value of a card
int crdval(int cards, int total) {
    // Ace
    if (cards == 0)
        return (total + 11 <= 21) ? 11 : 1;
    // 10, J, Q, K
    else if (cards >= 9)
        return 10;
    // 2 to 9
    else
        return cards + 1;
}

int main(int argc, char** argv) {
```

```
// Random number generator
srand(time(0));

// Contains the string values for the cards
string cards[13] = {"A", "2", "3", "4", "5", "6", "7", "8", "9", "10", "J", "Q", "K"};

// Starting menu
cout << "<><><><><><><><><><><><><><><><><>\n";
cout << "     Welcome to Liam's Casino!\n";
cout << "         - Blackjack -\n             $ $ $\n";
cout << "     Enter any key to begin:\n";
cout << "<><><><><><><><><><><><><><><><><>\n";
// Accepts any input from the keyboard
cin.get();

// Betting UI
cout << "--------------------------------\n";
cout << "           Bet Rates:\n";
cout << "       $ Win: 2x your bet\n";
cout << "    $$ Blackjack: 3x your bet\n";

// Array to store the player's cards
int pcard[10];
// Array to store the dealer's cards
int dcard[10];
// Starts player with $100
float bal = 100.00;
// Variable that controls whether or not the game continues
bool again = true;

// While the player wants to play again and balance is more than the minimum bet
while (again == true && bal > 5){

   // Blackjack flag
   bool bj = false;
   // Total value of cards the player has
   int ptotal = 0;
   // Total value of cards the dealer has
   int dtotal = 0;
   // Initialize pcards to 1
   int pcards = 1;
   // Initialize dcards to 1
   int dcards = 1;
```

```
// Generate two random cards for the player and dealer
for (int i = 0; i < 2; i++) {
   pcard[i] = rancrd();
   dcard[i] = rancrd();
}

// Calculate the initial total of the player's cards
for (int i = 0; i < 2; i++) {
   ptotal += crdval(pcard[i], ptotal);
   dtotal += crdval(dcard[i], dtotal);
}

// Initial balance
cout << "--------------------------------\n";
cout << "   - Your balance is: $"<< bal << " -\n\n";
cout << "        Place your bets!\n        Min Bet is $5.00  \n        Max Bet is $50.00  \n";
cout << "--------------------------------\n";

// Initial bet
float bet = 0;
cout << "Your bet: $";
cin >> bet;
cout << "--------------------------------\n";

// Makes sure bet is within range
while (bet < 5 or bet > 50){
   cout << "Please choose an amount that is between $5.00 and $50.00\n";
   cout << "Your bet: $";
   cin >> bet;
   cout << "--------------------------------\n";
}

// Display the player's cards
cout << "Your cards:\n\n";
for (int i = 0; i < 2; i++) {
   cout << "|" << cards[pcard[i]] << "| ";
}
// Displays the total
cout << "\n\nTotal: " << ptotal << endl;

// Check if the player has immediate Blackjack
if (ptotal == 21) {
   // float betadd to let the player know how much money they made
   float betadd = (bet * 3);
```

```
    cout << "-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-\n";
    cout << " Lucky you, first try Blackjack!\n";
    cout << "           You win!\n\n";
    cout << "         You made $" << betadd << "!\n";
    // Updating balance with 3x bet because it was a blackjack
    bal = ((bet * 3) + bal);
    // Formatting for $X.00
    cout << fixed << setprecision(2);
    cout << "  - Your balance is: $"<< bal << " -\n";
    cout << "-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-\n";
    cout << "           Play again?\nEnter 'Y' to continue or 'N' to exit.\n";
    char choice;
    cin >> choice;
    // Clear the input buffer
    cin.ignore();

    // Validate the input
    while (choice != 'Y' && choice != 'y' && choice != 'N' && choice != 'n') {
        cout << "Invalid choice, Enter 'Y' to continue or 'N' to exit.\n";
        cin >> choice;
        // Clear the input buffer
        cin.ignore();
    }
    // If the user chooses yes, the game repeats
    if (choice == 'y' or choice == 'Y'){
        again = true;
    }
    // If the user chooses no, the game ends
    else if(choice == 'n' or choice == 'N'){
        again = false;
    }

    bj = true;
}

char choice;
bool turn = true;

// Player's turn to hit or stay
while (ptotal < 21 && turn) {
    cout << "--------------------------------\n";
    cout << "Enter 'H' to hit or 'S' to stay.\n";
    cin >> choice;
    cin.ignore();
```

```
cout << "-------------------------------\n";

// If player decides to hit
if (choice == 'H' or choice == 'h') {
    pcards++;
    pcard[pcards] = rancrd();
    cout << "Your cards:\n\n";
    for (int i = 0; i <= pcards; i++) {
        cout << "|" << cards[pcard[i]] << "| ";
    }

    ptotal += crdval(pcard[pcards], ptotal);

    cout << "\n\nTotal: " << ptotal << endl;

    if (ptotal == 21) {
        float betadd = (bet * 3);
        cout << "-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-\n";
        cout << "       Blackjack! You win!\n\n";
        cout << "         You made $" << betadd << "!\n";
        // Updating balance with 3x bet because it was a blackjack
        bal = ((bet * 3) + bal);
        // Formatting for $X.00
        cout << fixed << setprecision(2);
        cout << "  - Your balance is: $"<< bal << " -\n";
        cout << "-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-\n";
        cout << "           Play again?\nEnter 'Y' to continue or 'N' to exit.\n";
        char choice;
        cin >> choice;
        // Clear the input buffer
        cin.ignore();

    // Validate the input
    while (choice != 'Y' && choice != 'y' && choice != 'N' && choice != 'n') {
        cout << "Invalid choice, Enter 'Y' to continue or 'N' to exit.\n";
        cin >> choice;
        // Clear the input buffer
        cin.ignore();
    }
    // If the user chooses yes, the game repeats
    if (choice == 'y' or choice == 'Y'){
        again = true;
    }
    // If the user chooses no, the game ends
    else if(choice == 'n' or choice == 'N'){
        again = false;
    }
        bj = true;
        turn = false;
    }

    else if (ptotal > 21) {
        cout << "-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-\n";
        cout << "        Bust! You lose.\n\n";
        cout << "         You lost $" << bet << "!\n";
        // Updating balance with - bet because the user lost
        bal = (bal - bet);
        // Formatting for $X.00
        cout << fixed << setprecision(2);
        cout << "  - Your balance is: $"<< bal << " -\n";
        cout << "-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-\n";
        cout << "            Play again?\nEnter 'Y' to continue or 'N' to exit.\n";
        char choice;
        cin >> choice;
        // Clear the input buffer
        cin.ignore();

        // Validate the input
        while (choice != 'Y' && choice != 'y' && choice != 'N' && choice != 'n') {
            cout << "Invalid choice, Enter 'Y' to continue or 'N' to exit.\n";
            cin >> choice;
            // Clear the input buffer
            cin.ignore();
        }
        // If the user chooses yes, the game repeats
        if (choice == 'y' or choice == 'Y'){
            again = true;
        }
        // If the user chooses no, the game ends
        else if(choice == 'n' or choice == 'N'){
            again = false;
        }
    bj = true;
    turn = false;
    }
}
else if (choice == 'S' or choice == 's') {
    cout << "       You chose to stay.\n";
```

```
        turn = false;
    }
    else {
        cout << "Invalid choice. Please enter 'H' to hit or 'S' to stay.\n";
    }
}

// Dealer's turn to hit
while (dtotal < 17) {
    dcards++;
    dcard[dcards] = rancrd();
    dtotal += crdval(dcard[dcards], dtotal);
}

//Shows this when player does not blackjack or bust
if (bj == false) {
    cout << "--------------------------------\n";
    cout << "Dealer's cards:\n\n";
    for (int i = 0; i <= dcards; i++) {
        cout << "|" << cards[dcard[i]] << "| ";
    }
    cout << "\n\nTotal: " << dtotal << "\n";

    //If player wins against the dealer
    if (dtotal > 21 or (ptotal <= 21 && ptotal > dtotal)) {
        float betadd = (bet * 2);
        cout << "-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-\n";
        cout << "   Congratulations! You win!\n\n";
        cout << "          You made $" << betadd << "!\n";
        // Adjusted for regular win (2x the winnings instead of 3x)
        bal = ((bet * 2) + bal);
        // Formatting for $X.00
        cout << fixed << setprecision(2);
        cout << "  - Your balance is: $"<< bal << " -\n";
        cout << "-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-\n";
        cout << "            Play again?\nEnter 'Y' to continue or 'N' to exit.\n";
        char choice;
        cin >> choice;
        // Clear the input buffer
        cin.ignore();

        // Validate the input
        while (choice != 'Y' && choice != 'y' && choice != 'N' && choice != 'n') {
            cout << "Invalid choice, Enter 'Y' to continue or 'N' to exit.\n";
        cin >> choice;
        // Clear the input buffer
        cin.ignore();
    }
    // If the user chooses yes, the game repeats
    if (choice == 'y' or choice == 'Y'){
        again = true;
    }
    // If the user chooses no, the game ends
    else if(choice == 'n' or choice == 'N'){
        again = false;
    }
}

// Tie between player and dealer
else if (ptotal == dtotal) {
    cout << "-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-\n";
    cout << "           It's a tie!\n";
    cout << "-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-\n";
    cout << "            Play again?\nEnter 'Y' to continue or 'N' to exit.\n";
    char choice;
    cin >> choice;
    // Clear the input buffer
    cin.ignore();

    // Validate the input
    while (choice != 'Y' && choice != 'y' && choice != 'N' && choice != 'n') {
        cout << "Invalid choice, Enter 'Y' to continue or 'N' to exit.\n";
        cin >> choice;
        // Clear the input buffer
        cin.ignore();
    }
    // If the user chooses yes, the game repeats
    if (choice == 'y' or choice == 'Y'){
        again = true;
    }
    // If the user chooses no, the game ends
    else if(choice == 'n' or choice == 'N'){
        again = false;
    }
}

// Dealer wins
else {
```

```
            cout << "-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-\n";
            cout << "        The dealer wins.\n\n";
            cout << "         You lost $" << bet << "!\n";
            // Subtracts the bet since the user lost
            bal = (bal - bet);
            // Formatting for $X.00
            cout << fixed << setprecision(2);
            cout << "  - Your balance is: $"<< bal << " -\n";
            cout << "-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-\n";
            cout << "           Play again?\nEnter 'Y' to continue or 'N' to exit.\n";
            char choice;
            cin >> choice;
            // Clear the input buffer
            cin.ignore();

            // Validate the input
            while (choice != 'Y' && choice != 'y' && choice != 'N' && choice != 'n') {
                cout << "Invalid choice, Enter 'Y' to continue or 'N' to exit.\n";
                cin >> choice;
                // Clear the input buffer
                cin.ignore();
            }
            // If the user chooses yes, the game repeats
            if (choice == 'y' or choice == 'Y'){
                again = true;
            }
            // If the user chooses no, the game ends
            else if(choice == 'n' or choice == 'N'){
                again = false;
            }
        }
    }
        // If your balance is less than the minimum bet, then the program shuts down
        if (bal < 5){
            cout << "Sorry! Your balance is lower than the minimum bet.";
        }
    }

    //Exit Program
    return 0;
}
```