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."

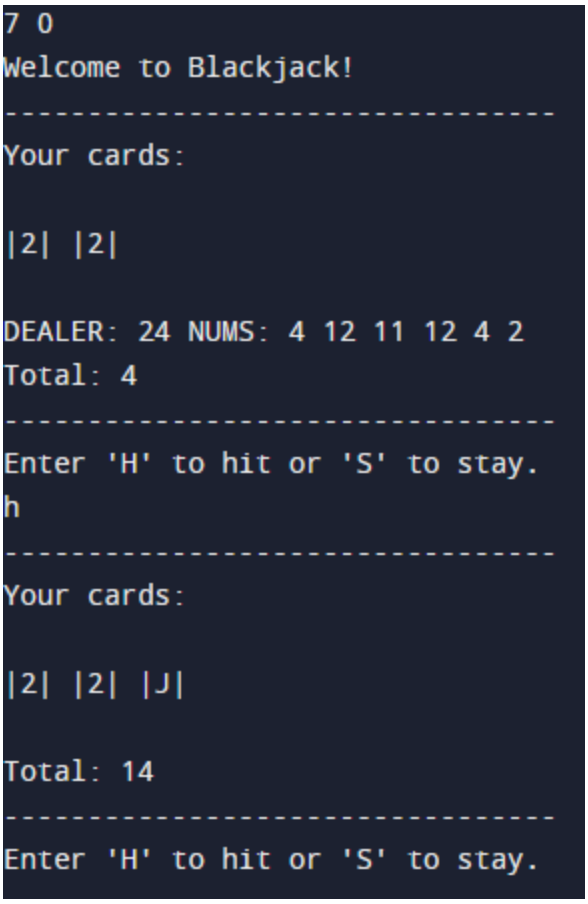
In blackjack each card has a value equivalent to its face value except for face cards (Kings, Queens and Jacks) which are worth 10 points. The Ace can be counted as either 1 or 11 depending on how it suits the players strategy. The dealer follows rules when drawing cards and players must make strategic 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	**270** - (69.33%)
Comment Lines	**81** - (20.88%)
Blank Lines	**38** - (9.79%)
Total Lines	**389**

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

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.

```
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.
```

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

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.

```
<><><><><><><><><><><><><><><><><><>
     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.
```

V1.3: UI and Gambling

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.

V1.4: Removing Global Variables:

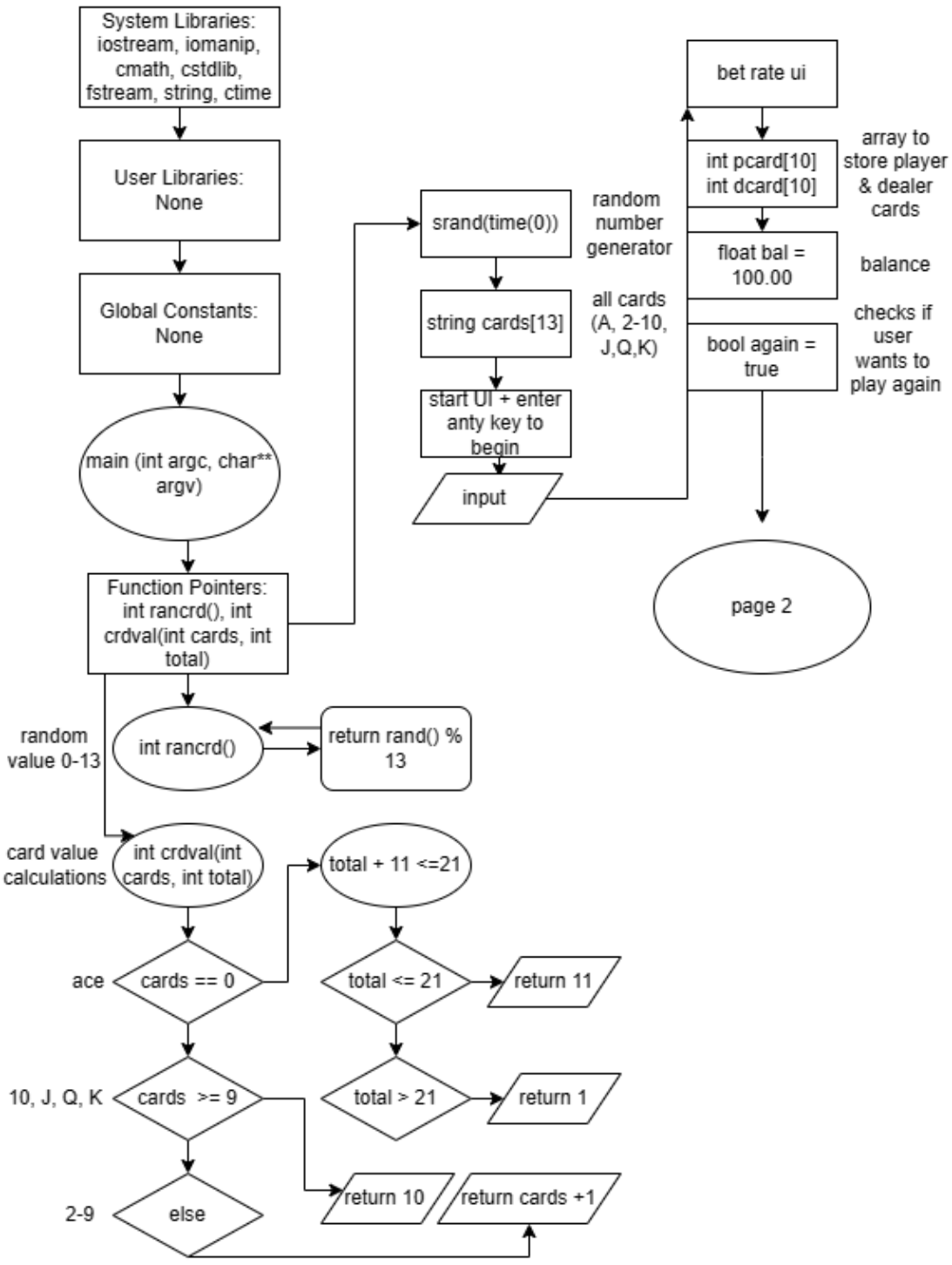
Version 1.4 of the project aimed primarily to remove the reliance on global variables, particularly 'int rancrd' and 'int crdval,' by relocating them within the main function. The strategy involved moving the entire functions into the main function and adapting their formatting to use pointers, which streamlined the process and minimized alterations.

Additionally, a bug was identified in a while loop responsible for prompting the user to place a bet within the $5-$50 range. The loop was getting stuck in an infinite loop. To address this issue, I implemented an input clearing mechanism at the loop's end, preventing it from incorrectly detecting that the user's input was out of range and resolving the problem.

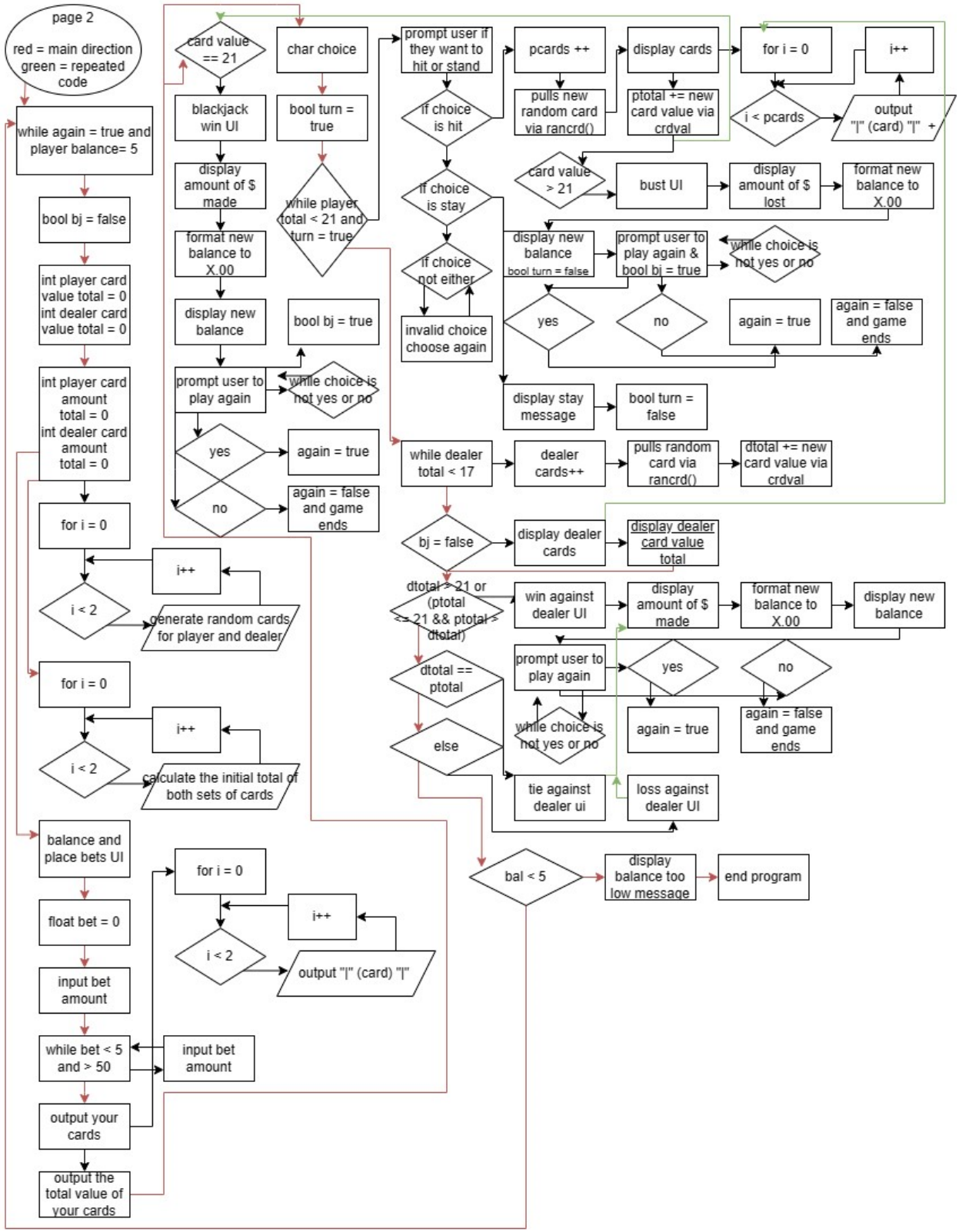
Flowchart Page 1

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

Project 1: Blackjack



Flowchart Page 2



Pseudo Code

```
Include <iostream>, <iomanip>, <cmath>, <cstdlib>, <fstream>, <string>, and <ctime> libraries

Main Function

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

// Function pointer to get total of cards
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)

   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 repositiory

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

```
1.  /*
2.   * File:   main.cpp
3.   * Author: Liam Shaw
4.   *
5.   * Created on July, 22
6.   * Purpose: The classic card game, Blackjack.
7.   */
8.
9.  //System Libraries
10. #include <iostream>
11. #include <iomanip>
12. #include <cmath>
13. #include <cstdlib>
14. #include <fstream>
15. #include <string>
16. #include <ctime>
17.
18. using namespace std;
19.
20. int main(int argc, char** argv) {
21.
22.     // Function pointer to generate a random card index from 0 to 12 (representing A to K)
23.     int (*rancrd)() = []() -> int  {
24.         return rand() % 13;
25.     };
26.
27.     // Function pointer to get the value of a card
28.     int (*crdval)(int, int) = [](int cards, int total) -> int  {
29.         // Ace
30.         if (cards == 0)
31.             return (total + 11 <= 21) ? 11 : 1;
32.         // 10, J, Q, K
33.         else if (cards >= 9)
34.             return 10;
35.         // 2 to 9
36.         else
37.             return cards + 1;
38.     };
39.
40.     // Random number generator
41.     srand(time(0));
42.
43.     // Contains the string values for the cards
44.     string cards[13] = {"A", "2", "3", "4", "5", "6", "7", "8", "9", "10", "J", "Q", "K"};
45.
46.     // Starting menu
47.     cout << "<><><><><><><><><><><><><><><><><><><><><><><>\n";
48.     cout << "     Welcome to Liam's Casino!\n";
49.     cout << "         - Blackjack -\n            $ $ $\n";
50.     cout << "     Enter any key to begin:\n";
51.     cout << "<><><><><><><><><><><><><><><><><><><><><><><>\n";
```

```
52.     // Accepts any input from the keyboard
53.     cin.get();
54.
55.     // Betting UI
56.     cout << "---------------------------------\n";
57.     cout << "           Bet Rates:\n";
58.     cout << "       $ Win: 2x your bet\n";
59.     cout << "    $$ Blackjack: 3x your bet\n";
60.
61.     // Array to store the player's cards
62.     int pcard[10];
63.     // Array to store the dealer's cards
64.     int dcard[10];
65.     // Starts player with $100
66.     float bal = 100.00;
67.     // Variable that controls whether or not the game continues
68.     bool again = true;
69.
70.     // While the player wants to play again and balance is more than the minimum bet
71.     while (again == true && bal > 5){
72.
73.         // Blackjack flag
74.         bool bj = false;
75.         // Total value of cards the player has
76.         int ptotal = 0;
77.         // Total value of cards the dealer has
78.         int dtotal = 0;
79.         // Initialize pcards to 1
80.         int pcards = 1;
81.         // Initialize dcards to 1
82.         int dcards = 1;
83.
84.     // Generate two random cards for the player and dealer
85.     for (int i = 0; i < 2; i++) {
86.         pcard[i] = rancrd();
87.         dcard[i] = rancrd();
88.     }
89.
90.     // Calculate the initial total of the player's cards
91.     for (int i = 0; i < 2; i++) {
92.         ptotal += crdval(pcard[i], ptotal);
93.         dtotal += crdval(dcard[i], dtotal);
94.     }
95.
96.     // Initial balance
97.     cout << "---------------------------------\n";
98.     cout << "   - Your balance is: $"<< bal << " -\n\n";
99.     cout << "        Place your bets!\n        Min Bet is $5.00  \n        Max Bet is $50.00  \n";
100.    cout << "---------------------------------\n";
101.
102.    // Initial bet
103.    float bet = 0;
104.    cout << "Your bet: $";
105.    cin >> bet;
```

```
106.    cout << "---------------------------------\n";
107.
108.    // Makes sure bet is within range
109.    while (bet < 5 or bet > 50){
110.        cout << "Please choose an amount that is between $5.00 and $50.00\n";
111.        cout << "Your bet: $";
112.        cin >> bet;
113.        cout << "---------------------------------\n";
114.        cin.clear();
115.    }
116.
117.    // Display the player's cards
118.    cout << "Your cards:\n\n";
119.    for (int i = 0; i < 2; i++) {
120.        cout << "|" << cards[pcard[i]] << "| ";
121.    }
122.    // Displays the total
123.    cout << "\n\nTotal: " << ptotal << endl;
124.
125.    // Check if the player has immediate Blackjack
126.    if (ptotal == 21) {
127.        // float betadd to let the player know how much money they made
128.        float betadd = (bet * 3);
129.        cout << "-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-\n";
130.        cout << " Lucky you, first try Blackjack!\n";
131.        cout << "            You win!\n\n";
132.        cout << "         You made $" << betadd << "!\n";
133.        // Updating balance with 3x bet because it was a blackjack
134.        bal = ((bet * 3) + bal);
135.        // Formatting for $X.00
136.        cout << fixed << setprecision(2);
137.        cout << "  - Your balance is: $"<< bal << " -\n";
138.        cout << "-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-\n";
139.        cout << "           Play again?\nEnter 'Y' to continue or 'N' to exit.\n";
140.        char choice;
141.        cin >> choice;
142.        // Clear the input buffer
143.        cin.ignore();
144.
145.        // Validate the input
146.        while (choice != 'Y' && choice != 'y' && choice != 'N' && choice != 'n') {
147.            cout << "Invalid choice, Enter 'Y' to continue or 'N' to exit.\n";
148.            cin >> choice;
149.            // Clear the input buffer
150.            cin.ignore();
151.        }
152.        // If the user chooses yes, the game repeats
153.        if (choice == 'y' or choice == 'Y'){
154.            again = true;
155.        }
156.        // If the user chooses no, the game ends
157.        else if(choice == 'n' or choice == 'N'){
158.            again = false;
159.        }
```

```
160.
161.        bj = true;
162.    }
163.
164.    char choice;
165.    bool turn = true;
166.
167.    // Player's turn to hit or stay
168.    while (ptotal < 21 && turn) {
169.        cout << "---------------------------------\n";
170.        cout << "Enter 'H' to hit or 'S' to stay.\n";
171.        cin >> choice;
172.        cin.ignore();
173.        cout << "---------------------------------\n";
174.
175.        // If player decides to hit
176.        if (choice == 'H' or choice == 'h') {
177.            pcards++;
178.            pcard[pcards] = rancrd();
179.            cout << "Your cards:\n\n";
180.            for (int i = 0; i <= pcards; i++) {
181.                cout << "|" << cards[pcard[i]] << "| ";
182.            }
183.
184.            ptotal += crdval(pcard[pcards], ptotal);
185.
186.            cout << "\n\nTotal: " << ptotal << endl;
187.
188.            if (ptotal == 21) {
189.                float betadd = (bet * 3);
190.                cout << "-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-\n";
191.                cout << "       Blackjack! You win!\n\n";
192.                cout << "         You made $" << betadd << "!\n";
193.                // Updating balance with 3x bet because it was a blackjack
194.                bal = ((bet * 3) + bal);
195.                // Formatting for $X.00
196.                cout << fixed << setprecision(2);
197.                cout << "  - Your balance is: $"<< bal << " -\n";
198.                cout << "-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-\n";
199.                cout << "           Play again?\nEnter 'Y' to continue or 'N' to exit.\n";
200.                char choice;
201.                cin >> choice;
202.                // Clear the input buffer
203.                cin.ignore();
204.
205.            // Validate the input
206.            while (choice != 'Y' && choice != 'y' && choice != 'N' && choice != 'n') {
207.                cout << "Invalid choice, Enter 'Y' to continue or 'N' to exit.\n";
208.                cin >> choice;
209.                // Clear the input buffer
210.                cin.ignore();
211.            }
212.            // If the user chooses yes, the game repeats
213.            if (choice == 'y' or choice == 'Y'){
```

```
214.             again = true;
215.         }
216.         // If the user chooses no, the game ends
217.         else if(choice == 'n' or choice == 'N'){
218.             again = false;
219.         }
220.             bj = true;
221.             turn = false;
222.         }
223.
224.         else if (ptotal > 21) {
225.             cout << "-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-\n";
226.             cout << "        Bust! You lose.\n\n";
227.             cout << "         You lost $" << bet << "!\n";
228.             // Updating balance with - bet because the user lost
229.             bal = (bal - bet);
230.             // Formatting for $X.00
231.             cout << fixed << setprecision(2);
232.             cout << "  - Your balance is: $"<< bal << " -\n";
233.             cout << "-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-\n";
234.             cout << "           Play again?\nEnter 'Y' to continue or 'N' to exit.\n";
235.             char choice;
236.             cin >> choice;
237.             // Clear the input buffer
238.             cin.ignore();
239.
240.             // Validate the input
241.             while (choice != 'Y' && choice != 'y' && choice != 'N' && choice != 'n') {
242.                 cout << "Invalid choice, Enter 'Y' to continue or 'N' to exit.\n";
243.                 cin >> choice;
244.                 // Clear the input buffer
245.                 cin.ignore();
246.             }
247.             // If the user chooses yes, the game repeats
248.             if (choice == 'y' or choice == 'Y'){
249.                 again = true;
250.             }
251.             // If the user chooses no, the game ends
252.             else if(choice == 'n' or choice == 'N'){
253.                 again = false;
254.             }
255.         bj = true;
256.         turn = false;
257.         }
258.     }
259.     else if (choice == 'S' or choice == 's') {
260.         cout << "       You chose to stay.\n";
261.         turn = false;
262.     }
263.     else {
264.         cout << "Invalid choice. Please enter 'H' to hit or 'S' to stay.\n";
265.     }
266. }
267.
```

```
268.    // Dealer's turn to hit
269.    while (dtotal < 17) {
270.        dcards++;
271.        dcard[dcards] = rancrd();
272.        dtotal += crdval(dcard[dcards], dtotal);
273.    }
274.
275.    //Shows this when player does not blackjack or bust
276.    if (bj == false) {
277.        cout << "---------------------------------\n";
278.        cout << "Dealer's cards:\n\n";
279.        for (int i = 0; i <= dcards; i++) {
280.            cout << "|" << cards[dcard[i]] << "| ";
281.        }
282.        cout << "\n\nTotal: " << dtotal << "\n";
283.
284.        //If player wins against the dealer
285.        if (dtotal > 21 or (ptotal <= 21 && ptotal > dtotal)) {
286.            float betadd = (bet * 2);
287.            cout << "-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-\n";
288.            cout << "   Congratulations! You win!\n\n";
289.            cout << "         You made $" << betadd << "!\n";
290.            // Adjusted for regular win (2x the winnings instead of 3x)
291.            bal = ((bet * 2) + bal);
292.            // Formatting for $X.00
293.            cout << fixed << setprecision(2);
294.            cout << "  - Your balance is: $"<< bal << " -\n";
295.            cout << "-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-\n";
296.            cout << "           Play again?\nEnter 'Y' to continue or 'N' to exit.\n";
297.            char choice;
298.            cin >> choice;
299.            // Clear the input buffer
300.            cin.ignore();
301.
302.            // Validate the input
303.            while (choice != 'Y' && choice != 'y' && choice != 'N' && choice != 'n') {
304.                cout << "Invalid choice, Enter 'Y' to continue or 'N' to exit.\n";
305.                cin >> choice;
306.                // Clear the input buffer
307.                cin.ignore();
308.            }
309.            // If the user chooses yes, the game repeats
310.            if (choice == 'y' or choice == 'Y'){
311.                again = true;
312.            }
313.            // If the user chooses no, the game ends
314.            else if(choice == 'n' or choice == 'N'){
315.                again = false;
316.            }
317.        }
318.
319.        // Tie between player and dealer
320.        else if (ptotal == dtotal) {
321.            cout << "-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-\n";
```

```
322.            cout << "          It's a tie!\n";
323.            cout << "-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-\n";
324.            cout << "           Play again?\nEnter 'Y' to continue or 'N' to exit.\n";
325.            char choice;
326.            cin >> choice;
327.            // Clear the input buffer
328.            cin.ignore();
329.
330.            // Validate the input
331.            while (choice != 'Y' && choice != 'y' && choice != 'N' && choice != 'n') {
332.                cout << "Invalid choice, Enter 'Y' to continue or 'N' to exit.\n";
333.                cin >> choice;
334.                // Clear the input buffer
335.                cin.ignore();
336.            }
337.            // If the user chooses yes, the game repeats
338.            if (choice == 'y' or choice == 'Y'){
339.                again = true;
340.            }
341.            // If the user chooses no, the game ends
342.            else if(choice == 'n' or choice == 'N'){
343.                again = false;
344.            }
345.        }
346.
347.        // Dealer wins
348.        else {
349.            cout << "-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-\n";
350.            cout << "        The dealer wins.\n\n";
351.            cout << "         You lost $" << bet << "!\n";
352.            // Subtracts the bet since the user lost
353.            bal = (bal - bet);
354.            // Formatting for $X.00
355.            cout << fixed << setprecision(2);
356.            cout << "  - Your balance is: $"<< bal << " -\n";
357.            cout << "-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-\n";
358.            cout << "           Play again?\nEnter 'Y' to continue or 'N' to exit.\n";
359.            char choice;
360.            cin >> choice;
361.            // Clear the input buffer
362.            cin.ignore();
363.
364.            // Validate the input
365.            while (choice != 'Y' && choice != 'y' && choice != 'N' && choice != 'n') {
366.                cout << "Invalid choice, Enter 'Y' to continue or 'N' to exit.\n";
367.                cin >> choice;
368.                // Clear the input buffer
369.                cin.ignore();
370.            }
371.            // If the user chooses yes, the game repeats
372.            if (choice == 'y' or choice == 'Y'){
373.                again = true;
374.            }
375.            // If the user chooses no, the game ends
```

```
376.            else if(choice == 'n' or choice == 'N'){
377.                again = false;
378.            }
379.        }
380.    }
381.        // If your balance is less than the minimum bet, then the program shuts down
382.        if (bal < 5){
383.            cout << "Sorry! Your balance is lower than the minimum bet.";
384.        }
385.    }
386.
387.    //Exit Program
388.    return 0;
389.}
```