Project 2

Title

Blackjack

Course

CIS-5

Section

45428

Due Date

July 30, 2023

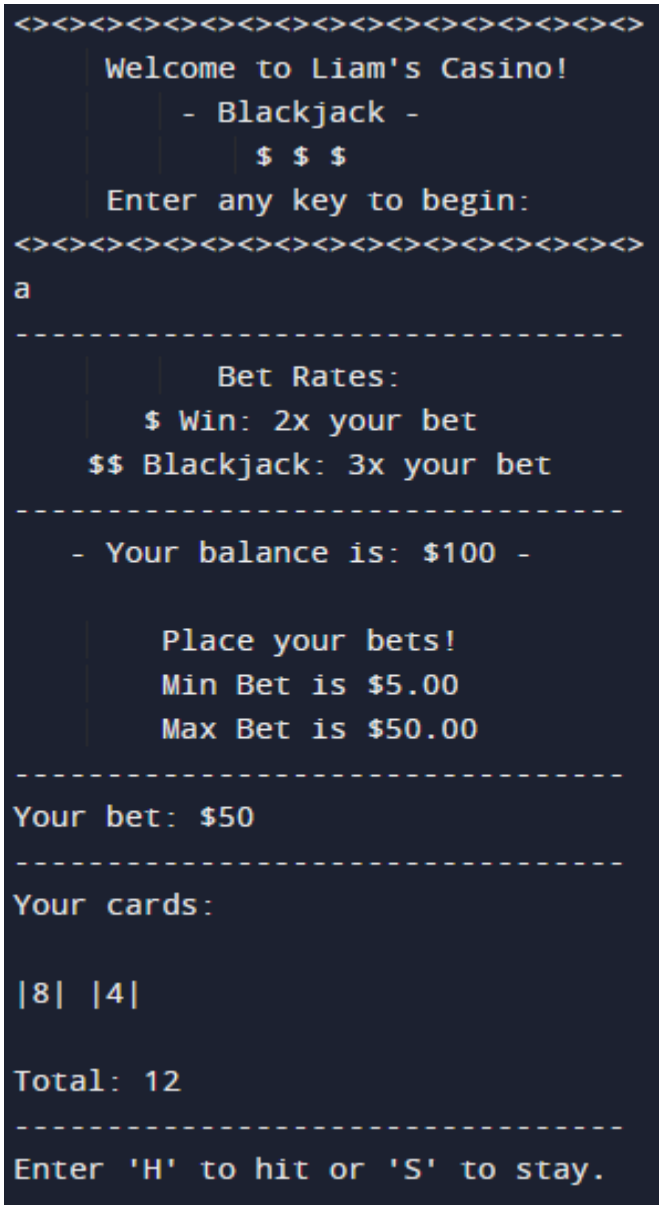
Author

Liam Shaw

TABLE OF CONTENTS

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.



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."

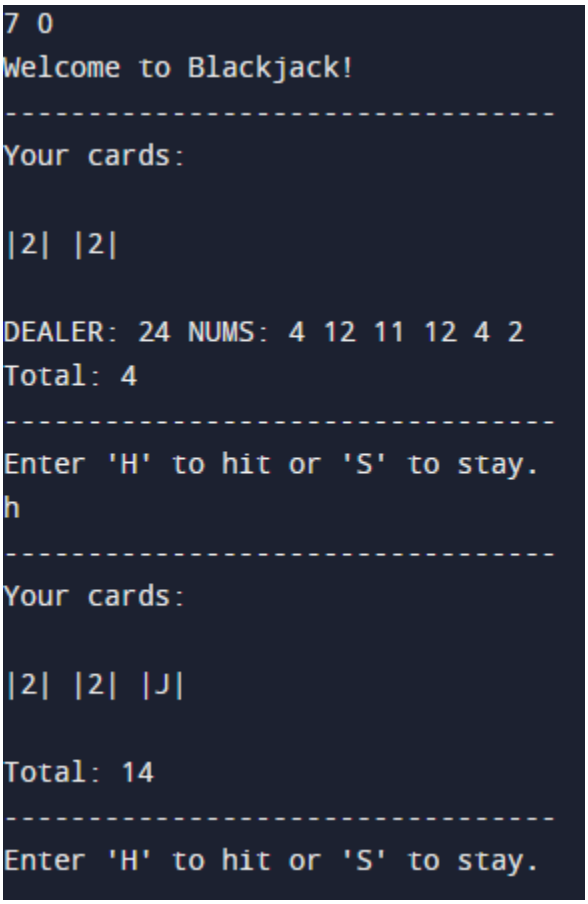
In blackjack each card has a value equivalent to its face value except for face cards (Kings, Queens and Jacks) which are worth 10 points. The Ace can be counted as either 1 or 11 depending on how it suits the players strategy. The dealer follows rules when drawing cards and players must make strategic 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	**187** - (70.30%)
Comment Lines	**42** - (15.79%)
Blank Lines	**37** - (13.91%)
Total Lines	**266**

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>, <ctime>, and <vector>.

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
--------------------------------
```

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.

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.

V1.5: Adding Proper Functions

Version 1.4 of the game contained the entire game logic within the main function, resulting in a lengthy and intricate code block. Firstly, I introduced two new functions: generateRandomCard(), which generates a random card index, and getCardValue(), responsible for determining the value of a card based on its index. These functions replaced the function pointer expressions present in the original code, streamlining the structure.

Additionally, I included a new function, displayCards(), which displays the cards in a neatly formatted manner. Then, I created a function called playRound() that encapsulates the logic for a single round of Blackjack. This function takes the player's balance as a reference, allowing it to update the balance as the game progresses. To eliminate unnecessary variables, I moved the game-specific variables, like player and dealer card arrays, their total values, and the Blackjack flag, into the playRound() function.

Now, the game's flow resides within the playRound() function, while the main function handles the main loop, enabling the player to participate in multiple rounds until their balance falls below the minimum bet or they choose to exit. As a result of these changes, the main function now only calls playRound() and manages the loop and balance. The game still functions the exact same, negating the need for screenshots

V1.6: Implementing Vectors, etc.

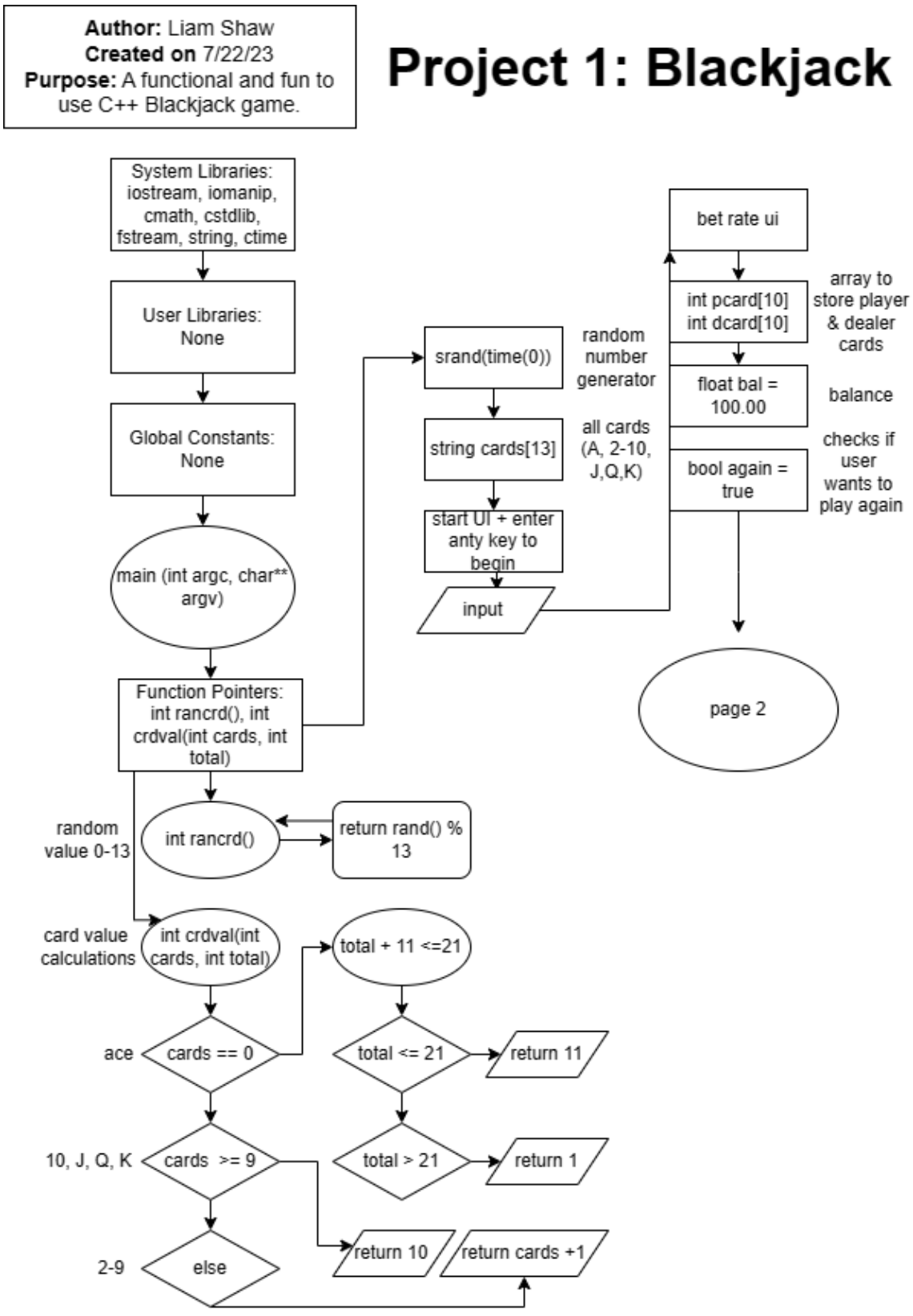
In the improved version of the Blackjack game code, I made various enhancements to its functionality and user experience. Firstly, I replaced fixed-size arrays with dynamic vectors from the <vector> library to store the player's and dealer's cards. This change allows the program to add cards dynamically to the hands, removing any fixed-size limitations. The player's and dealer's hands are now represented by vector<int> pcard and vector<int> dcard, respectively.

To improve the display of cards, I modified the displayCards function to accept vectors as parameters. By utilizing a range-based for loop, the function can now display all the cards present in the vectors effectively.

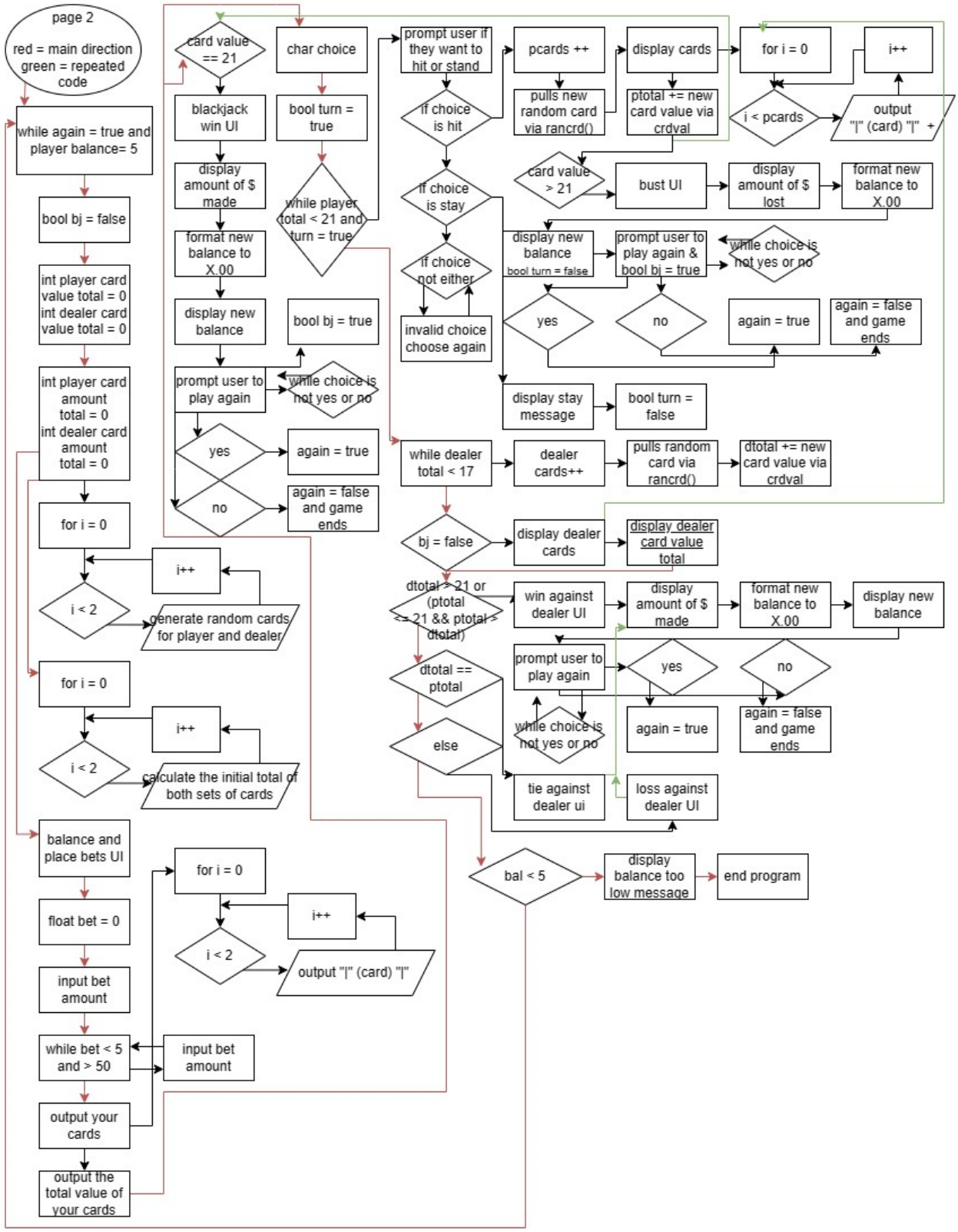
Another significant improvement was introducing function overloading for the getCardValue function. The first version of getCardValue now takes an extra parameter called aceAsEleven, allowing players to choose whether an Ace should be valued as 1 or 11. To maintain the default behavior of valuing Aces as 11, I created a second version of getCardValue as an overloaded function without the aceAsEleven parameter. This overloaded version internally calls the first version with aceAsEleven set to true.

To make the card value calculation more flexible, I replaced the existing getCardValue calls in the playRound function with the newly introduced overloaded version. This change empowers players with the option to determine how Aces are valued in their game, significantly enhancing the overall Blackjack experience.

Flowchart Page 1



Flowchart Page 2



Pseudo Code

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

Function generateRandomCard() -> Integer
   Return random number between 0 and 12

Function getCardValue(card: Integer, total: Integer, aceAsEleven: Boolean) -> Integer
   If card is 0 then
      If (total + 11 <= 21) AND aceAsEleven is true then
         Return 11
      Else
         Return 1
   Else If card is greater than or equal to 9 then
      Return 10
   Else
      Return card + 1

Function getCardValue(card: Integer, total: Integer) -> Integer
   Return getCardValue(card, total, true)

Function displayCards(cardVector: List of Integers, cards: List of Strings)
   For each card in cardVector do
      Display "|" + cards[card] + "| "
   End For
   Display newline

Function playRound(balance: Float) -> Boolean
   Initialize bj as false
   Initialize ptotal as 0
   Initialize dtotal as 0
   Initialize pcards as 1
   Initialize dcards as 1
   Initialize bet as 0

   Initialize cards as a List of Strings {"A", "2", "3", "4", "5", "6", "7", "8", "9", "10", "J", "Q", "K"}
   Initialize pcard as an empty List of Integers
   Initialize dcard as an empty List of Integers

   Display "---------------------------------"
   Display "          Bet Rates:"
   Display "      $ Win: 2x your bet"
   Display "   $$ Blackjack: 3x your bet"

   Repeat
      Repeat
         Display "---------------------------------"
         Display "   - Your balance is: $" + balance + " -"
         Display "        Place your bets!"
         Display "        Min Bet is $5.00"
         Display "        Max Bet is $50.00"
         Display "---------------------------------"
         Display "Your bet: $"
```

```
        Input bet
        Display "---------------------------------"
    Until bet >= 5 AND bet <= 50

    Clear pcard and dcard
    ptotal = 0
    dtotal = 0

    Repeat 2 times
        Add generateRandomCard() to pcard
        Add generateRandomCard() to dcard
    End Repeat

    Repeat 2 times
        ptotal += getCardValue(pcard[i], ptotal)
        dtotal += getCardValue(dcard[i], dtotal)
    End Repeat

    Display "Your cards:"
    DisplayCards(pcard, cards)
    Display "Total: " + ptotal

    If ptotal == 21 then
        betadd = bet * 3
        Display "-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-"
        Display " Lucky you, first try Blackjack!"
        Display "           You win!"
        Display "         You made $" + betadd + "!"
        balance += betadd
        Display "  - Your balance is: $" + balance + " -"
        Display "-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-"
        Set bj to true
    End If

    Set choice to ' '
    Set turn to true

    While ptotal < 21 AND turn is true
        Display "---------------------------------"
        Display "Enter 'H' to hit or 'S' to stay."
        Input choice
        If choice is 'H' OR choice is 'h' then
            pcards += 1
            Add generateRandomCard() to pcard
            Display "Your cards:"
            DisplayCards(pcard, cards)
            ptotal += getCardValue(pcard[pcards], ptotal)
            Display "Total: " + ptotal
            If ptotal == 21 then
                betadd = bet * 3
                Display "-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-"
                Display "       Blackjack! You win!"
                Display "         You made $" + betadd + "!"
                balance += betadd
```

```
            Display "  - Your balance is: $" + balance + " -"
            Display "-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-"
            Set bj to true
            Set turn to false
        Else If ptotal > 21 then
            Display "-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-"
            Display "        Bust! You lose."
            Display "         You lost $" + bet + "!"
            balance -= bet
            Display "  - Your balance is: $" + balance + " -"
            Display "-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-"
            Set bj to true
            Set turn to false
        End If
    Else If choice is 'S' OR choice is 's' then
        Display "       You chose to stay."
        Set turn to false
    Else
        Display "Invalid choice. Please enter 'H' to hit or 'S' to stay."
    End If
End While

While dtotal < 17 AND bj is false
    dcards += 1
    Add generateRandomCard() to dcard
    dtotal += getCardValue(dcard[dcards], dtotal)
End While

If bj is false then
    Display "---------------------------------"
    Display "Dealer's cards:"
    DisplayCards(dcard, cards)
    Display "Total: " + dtotal
    Display "---------------------------------"

    If dtotal > 21 OR (ptotal <= 21 AND ptotal > dtotal) then
        betadd = bet * 2
        Display "-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-"
        Display "   Congratulations! You win!"
        Display "         You made $" + betadd + "!"
        balance += betadd
        Display "  - Your balance is: $" + balance + " -"
        Display "-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-"
    Else If ptotal == dtotal then
        Display "-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-"
        Display "          It's a tie!"
        Display "-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-"
    Else
        Display "-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-"
        Display "        The dealer wins."
        Display "         You lost $" + bet + "!"
        balance -= bet
        Display "  - Your balance is: $" + balance + " -"
        Display "-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-"
```

```
            End If
        End If

        Display "          Play again?"
        Display "Enter 'Y' to continue or 'N' to exit."
        Input choice
        Validate choice to be 'Y' OR 'y' OR 'N' OR 'n'
    Until choice is not 'Y' AND choice is not 'y'

    If balance < 5 then
        Display "!-------------------------------------------------!"
        Display " Sorry! Your balance is lower than the minimum bet."
        Display "!-------------------------------------------------!"
    End If

    Return (choice is 'Y' OR choice is 'y')
End Function

Function main()
    balance = 100.00
    again = true

    Display "<><><><><><><><><><><><><><><><><><><><><><><><>"
    Display "     Welcome to Liam's Casino!"
    Display "         - Blackjack -"
    Display "             $ $ $"
    Display "     Enter any key to begin:"
    Display "<><><><><><><><><><><><><><><><><><><><><><><><>"
    Wait for user input

    Repeat
        again = playRound(balance)
    Until again is false OR balance <= 5

    Return 0
End Function
```

Cross Reference for Project 1

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*	

Cross Reference for Project 2

Chapter	Section	Topic	Where Line #"s	Pts	Notes
6		*Functions*			
	3	*Function Prototypes*	*21,26,39,47*	*4*	*Always use prototypes*
	5	*Pass by Value*	*28,31,40,41…*	*4*	
	8	*return*	*22,35,232*	*4*	*A value from a function*
	9	*returning boolean*	*47,232*	*4*	
	10	*Global Variables*		*XXX*	*Do not use global variables -100 pts*
	11	*Static variables*		*4*	
	12	*Defaulted arguments*		*4*	
	13	*Pass by reference*	*47*	*4*	
	14	*overloading*	*26,39*	*5*	
	15	*exit() function*		*4*	
7		*Arrays*			
	1-6	*Single Dimensioned Arrays*	*61,62*	*3*	
	7	*Parallel Arrays*	*61,62*	*2*	
	8	*Single Dimensioned as Function Arguments*	*39*	*2*	
	9	*2 Dimensioned Arrays*		*2*	*Emulate style in book/in class repositiory*
	12	*STL Vectors*		*2*	
		Passing Arrays to and from Functions	*39, 109,142…*	*5*	
		Passing Vectors to and from Functions	*44,66,67…*	*5*	
8		*Searching and Sorting Arrays*			

	3	*Bubble Sort*		*4*	
	3	*Selection Sort*		*4*	
	1	*Linear or Binary Search*		*4*	
			TOTAL: 70 PTS		

References

1. Dr. Lehr's Lectures & Lab
2. "Starting Out with C++: From Control Structures through Objects" Gaddis, Tony. 8th Edition. (Textbook)
3. cplusplus.com

Program

```
1.  /*
2.   * File:   main.cpp
3.   * Author: Liam Shaw
4.   *
5.   * Created on July 30, 2023
6.   * Purpose: The classic card game, Blackjack.
7.   */
8.
9.  // System Libraries
10. #include <iostream>
11. #include <iomanip>
12. #include <cmath>
13. #include <fstream>
14. #include <cstdlib>
15. #include <ctime>
16. #include <vector>
17.
18. using namespace std;
19.
20. // Function to generate a random card index from 0 to 12 (representing A to K)
21. int generateRandomCard() {
22.     return rand() % 13;
23. }
24.
25. // Function to get the value of a card
26. int getCardValue(int card, int total, bool aceAsEleven) {
27.     // Ace can have a value of 1 or 11, depending on the current total and user preference
28.     if (card == 0)
29.         return (total + 11 <= 21 && aceAsEleven) ? 11 : 1;
30.     // 10, J, Q, K all have a fixed value of 10
31.     else if (card >= 9)
32.         return 10;
33.     // 2 to 9 have their face value as their value
34.     else
35.         return card + 1;
36. }
37.
38. // Overload the getCardValue function to use the default behavior (Ace as 11)
39. int getCardValue(int card, int total) {
40.     return getCardValue(card, total, true);
41. }
42.
43. // Function to display cards from a vector
44. void displayCards(const vector<int>& cardVector, const string cards[]) {
45.     for (int card : cardVector) {
46.         cout << "|" << cards[card] << "| ";
47.     }
48.     cout << endl;
49. }
50.
51. // Function to play a single round of Blackjack
```

```
52.  bool playRound(float& balance) {
53.      // Seed the random number generator with the current time
54.      srand(time(0));
55.
56.      // Array to store card names
57.      string cards[13] = {"A", "2", "3", "4", "5", "6", "7", "8", "9", "10", "J", "Q", "K"};
58.
59.      // Display the bet rates
60.      cout << "--------------------------------\n";
61.      cout << "            Bet Rates:\n";
62.      cout << "       $ Win: 2x your bet\n";
63.      cout << "    $$ Blackjack: 3x your bet\n";
64.
65.      // Vectors to store player's and dealer's cards
66.      vector<int> pcard;
67.      vector<int> dcard;
68.
69.      // Blackjack flag to check if someone has already won with Blackjack
70.      bool bj = false;
71.
72.      // Sets player and dealer cards total to 0
73.      int ptotal = 0;
74.      int dtotal = 0;
75.
76.      // Initialize the number of cards for player and dealer to 1
77.      int pcards = 1;
78.      int dcards = 1;
79.
80.      // Generate two random cards for the player and dealer
81.      for (int i = 0; i < 2; i++) {
82.          pcard.push_back(generateRandomCard());
83.          dcard.push_back(generateRandomCard());
84.      }
85.
86.      // Calculate the initial total of the player's and dealer's cards using the overloaded function
87.      for (int i = 0; i < 2; i++) {
88.          ptotal += getCardValue(pcard[i], ptotal);
89.          dtotal += getCardValue(dcard[i], dtotal);
90.      }
91.
92.      // Display initial balance and get the player's bet
93.      cout << "--------------------------------\n";
94.      cout << "   - Your balance is: $" << balance << " -\n\n";
95.      cout << "        Place your bets!\n        Min Bet is $5.00  \n        Max Bet is $50.00  \n";
96.      cout << "--------------------------------\n";
97.
98.      float bet = 0;
99.      cout << "Your bet: $";
100.     cin >> bet;
101.     cout << "--------------------------------\n";
102.
103.     // Makes sure bet is within range
104.     while (bet < 5 or bet > 50) {
105.         cout << "Please choose an amount that is between $5.00 and $50.00\n";
```

```
106.        cout << "Your bet: $";
107.        cin >> bet;
108.        cout << "---------------------------------\n";
109.        cin.clear();
110.    }
111.
112.    // Display the player's cards
113.    cout << "Your cards:\n\n";
114.    displayCards(pcard, cards);
115.    cout << "\nTotal: " << ptotal << endl;
116.
117.    // Check if the player has immediate Blackjack
118.    if (ptotal == 21) {
119.        float betadd = (bet * 3);
120.        cout << "-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-\n";
121.        cout << " Lucky you, first try Blackjack!\n";
122.        cout << "           You win!\n\n";
123.        cout << "         You made $" << betadd << "!\n";
124.        balance = ((bet * 3) + balance);
125.        cout << fixed << setprecision(2);
126.        cout << "  - Your balance is: $" << balance << " -\n";
127.        cout << "-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-\n";
128.        bj = true;
129.    }
130.
131.    char choice;
132.    bool turn = true;
133.
134.    // Player's turn to hit or stay
135.    while (ptotal < 21 && turn) {
136.        cout << "---------------------------------\n";
137.        cout << "Enter 'H' to hit or 'S' to stay.\n";
138.        cin >> choice;
139.        cin.ignore();
140.
141.        cout << "---------------------------------\n";
142.
143.        if (choice == 'H' or choice == 'h') {
144.            pcards++;
145.            pcard.push_back(generateRandomCard());
146.            cout << "Your cards:\n\n";
147.            displayCards(pcard, cards);
148.
149.            // Use the overloaded function here
150.            ptotal += getCardValue(pcard[pcards], ptotal);
151.
152.            cout << "\nTotal: " << ptotal << endl;
153.
154.            if (ptotal == 21) {
155.                float betadd = (bet * 3);
156.                cout << "-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-\n";
157.                cout << "       Blackjack! You win!\n\n";
158.                cout << "         You made $" << betadd << "!\n";
159.                balance = ((bet * 3) + balance);
```

```
160.                cout << fixed << setprecision(2);
161.                cout << "  - Your balance is: $" << balance << " -\n";
162.                cout << "-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-\n";
163.                bj = true;
164.                turn = false;
165.            } else if (ptotal > 21) {
166.                cout << "-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-\n";
167.                cout << "        Bust! You lose.\n\n";
168.                cout << "         You lost $" << bet << "!\n";
169.                balance = (balance - bet);
170.                cout << fixed << setprecision(2);
171.                cout << "  - Your balance is: $" << balance << " -\n";
172.                cout << "-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-\n";
173.                bj = true;
174.                turn = false;
175.            }
176.        } else if (choice == 'S' or choice == 's') {
177.            cout << "       You chose to stay.\n";
178.            turn = false;
179.        } else {
180.            cout << "Invalid choice. Please enter 'H' to hit or 'S' to stay.\n";
181.        }
182.    }
183.
184.    // Dealer's turn to hit
185.    while (dtotal < 17 && !bj) {
186.        dcards++;
187.        dcard.push_back(generateRandomCard());
188.        // Use the overloaded function here
189.        dtotal += getCardValue(dcard[dcards], dtotal);
190.    }
191.
192.    // Determine the winner
193.    if (!bj) {
194.        // Show the dealer's cards
195.        cout << "---------------------------------\n";
196.        cout << "Dealer's cards:\n\n";
197.        displayCards(dcard, cards);
198.        cout << "\nTotal: " << dtotal << "\n";
199.        cout << "---------------------------------\n";
200.
201.        if (dtotal > 21 || (ptotal <= 21 && ptotal > dtotal)) {
202.            float betadd = (bet * 2);
203.            cout << "-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-\n";
204.            cout << "   Congratulations! You win!\n\n";
205.            cout << "         You made $" << betadd << "!\n";
206.            balance = ((bet * 2) + balance);
207.            cout << fixed << setprecision(2);
208.            cout << "  - Your balance is: $" << balance << " -\n";
209.            cout << "-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-\n";
210.        } else if (ptotal == dtotal) {
211.            cout << "-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-\n";
212.            cout << "          It's a tie!\n";
213.            cout << "-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-\n";
```

```
214.        } else {
215.            cout << "-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-\n";
216.            cout << "        The dealer wins.\n\n";
217.            cout << "         You lost $" << bet << "!\n";
218.            balance = (balance - bet);
219.            cout << fixed << setprecision(2);
220.            cout << "  - Your balance is: $" << balance << " -\n";
221.            cout << "-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-\n";
222.        }
223.    }
224.
225.    // Ask if the player wants to play again
226.    cout << "           Play again?\nEnter 'Y' to continue or 'N' to exit.\n";
227.    cin >> choice;
228.    // Clear the input buffer
229.    cin.ignore();
230.
231.    // Validate the input
232.    while (choice != 'Y' && choice != 'y' && choice != 'N' && choice != 'n') {
233.        cout << "Invalid choice, Enter 'Y' to continue or 'N' to exit.\n";
234.        cin >> choice;
235.        // Clear the input buffer
236.        cin.ignore();
237.    }
238.
239.    return (choice == 'y' || choice == 'Y');
240.}
241.
242.int main(int argc, char** argv) {
243.    float balance = 100.00;
244.    bool again = true;
245.
246.    // Display welcome message and wait for player to press any key to start
247.    cout << "<><><><><><><><><><><><><><><><><><><><><><>\n";
248.    cout << "     Welcome to Liam's Casino!\n";
249.    cout << "          - Blackjack -\n             $ $ $\n";
250.    cout << "     Enter any key to begin:\n";
251.    cout << "<><><><><><><><><><><><><><><><><><><><><><>\n";
252.    cin.get();
253.
254.    // Play rounds of Blackjack until the player chooses not to continue or runs out of balance
255.    while (again && balance > 5) {
256.        again = playRound(balance);
257.        if (balance < 5) {
258.            cout << "\n!-------------------------------------------------!\n";
259.            cout << " Sorry! Your balance is lower than the minimum bet.\n";
260.            cout << "!-------------------------------------------------!\n";
261.        }
262.    }
263.
264.    // Exit Program
265.    return 0;
266.}
```