Project 1

Title

Blackjack

Course

CIS-17A

Section

42474

Due Date

4/22/2018

Author

Seth Tyler

Table of Contents

Introduction

Blackjack is a famous card-based game found in casinos and other gambling environments. It is one of the simplest games to grasp, which was ultimately the motivation for this program.

This game is completely text-based. While a version with GUI's would be more appealing and enticing to play, this game captures the gameplay sufficiently, while leaving out more complex features of the game such as doubling, splitting, and insurance.

Summary

This program is 413 lines long total. This includes header files such as Card.h and Deck.h

The program includes 10 functions, 9 system libraries, 2 structure files, and 18 variables (not including for-loop initializers.)

Overall, the game was more challenging to create than expected, even without the more complex features mentioned previously. The most challenging aspects were fulfilling the checklist and ensuring every chapter section was included in the code. While not everything was fulfilled, the game works beautifully and uses what is necessary.

In total, time spent on this project was roughly 10 hours. Most of this time was dedicated to fixing bugs and polishing the game, as well as catching up on knowledge that was not as fluent.

Objective

The objective of the game is to make the sum of the cards in your hand as close as possible to 21 without exceeding 21. Face cards are worth 10, aces are worth 11 or 1 (depending on the player's needs.)

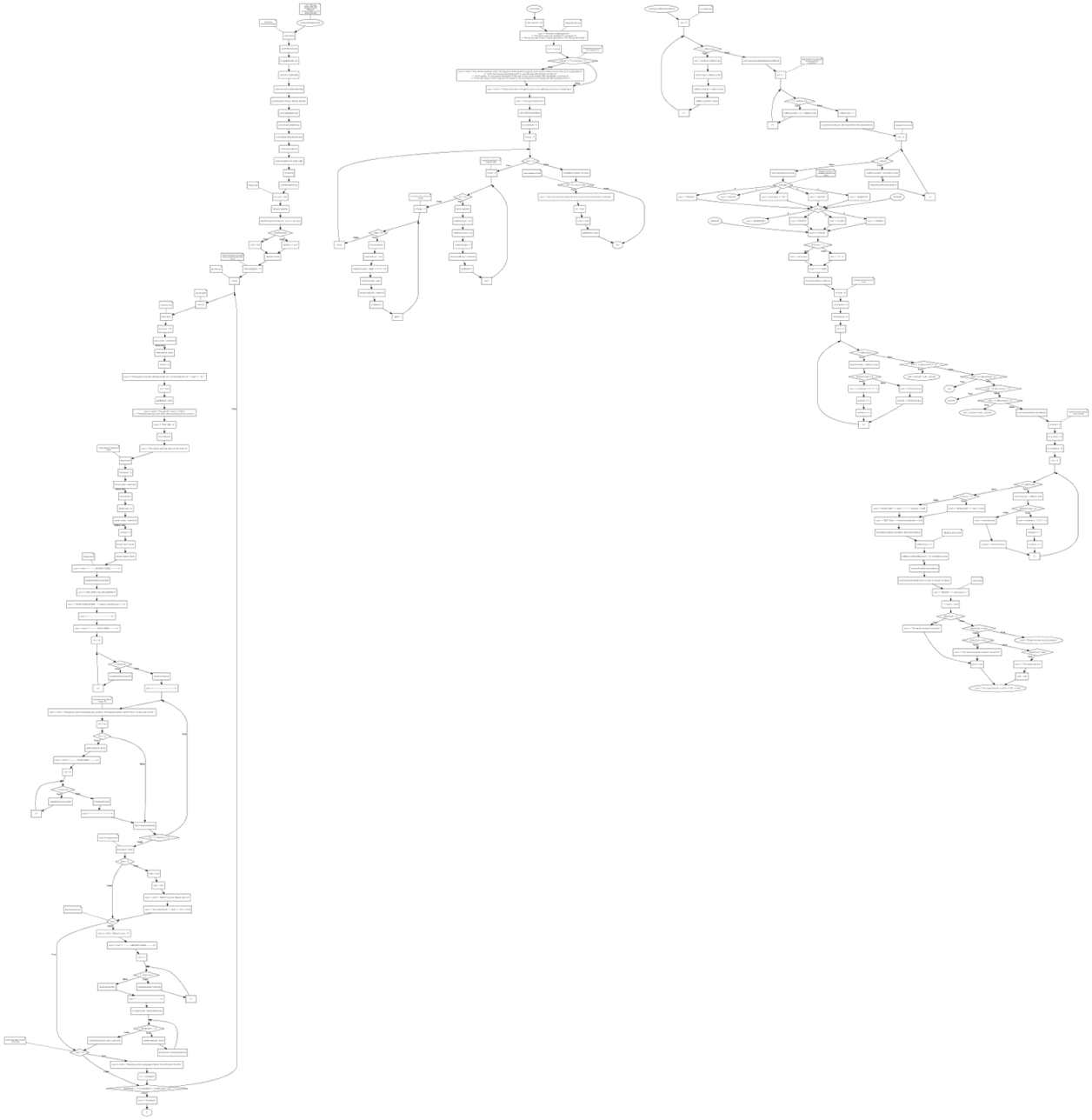
The game begins by placing your bet. The dealer will then deal you two cards face up and two cards to themselves, with one face down and one face up. You must determine if you would like another card (hit) or if you are comfortable with the amount that you have (stay)

If the dealer has a greater sum than you, or you exceed 21, they will win. If the dealer exceeds 21 (busts) or has a lower amount than you, you will win.

Sample Input/Output

```
Welcome to Blackjack!
This game uses one standard 52 card deck.
Would you like to learn how to play? Enter Y for YES and N for NO: n

Please note that in this game, there is no splitting, insurance, or doubling.
Let's get started.

Please place your bet. (Minimum bet: $6 | Current balance: $150.00 : 10

You bet $10.00
The dealer will now shuffle the deck and deal out the cards.
Shuffling...
The dealer will now deal out the cards.

---------- DEALER'S CARDS ----------
4 OF DIAMONDS (VALUE: 4)
ONE CARD STILL FACE DOWN
TOTAL SUM (SO FAR): 4
------------------------------------

---------- YOUR CARDS ----------
7 OF HEARTS (VALUE: 7)
JACK OF DIAMONDS (VALUE: 10)
TOTAL SUM: 17
BEST SUM: 17
--------------------------------

Would you like to stay (keep your cards) or hit (request another card)? Press 1 to stay and 2 to hit: 1

Dealer's turn...

---------- DEALER'S CARDS ----------
4 OF DIAMONDS (VALUE: 4)
ACE OF SPADES (VALUE: 1 / 11)
TOTAL SUM: 15 / 5
BEST SUM: 15
------------------------------------
DEALER: 15; YOU: 17
You have the better number. You win!
You now have $160.00

Would you like to play again? Enter Y for YES and N for NO: n
Goodbye!
RUN SUCCESSFUL (total time: 7s)
```

Flowchart



Variables

Variable	Description	Location
cash	Contain's the player's cash	43
dataFile	fstream to read in player's cash from money.txt	45
playAgain	Determines whether the player wants to play another round	55
hit	Determines if the player wants to hit or stay	63
bet	Contains the player's bet amount	71
hand	A Deck variable for the player's hand	84
dealer	A Deck variable for the dealer's hand	88
best	Contains the best sum of the player's hand	92
bust	Flag variable for if the player's sum exceeded 21. Round is over and player loses their money if true	130
tutorial	Determines if the player wants to know how to play	172
cardNum	An iterator for each of the 52 cards to be added into the deck	193
sum	The first sum of cards in a hand	298/326
aceSum	The second sum of cards in a hand that is shown if there is an ace in a hand	299/327
numAces	The number of aces in a hand	300/328

Checklist

Chapter	Section	Concept	Location in Code	Comments
9		**Pointers/Memory Allocation**		
	1	Memory Addresses	218	
	2	Pointer Variables	192	
	3	Arrays/Pointers	192, 202, Deck.h	
	4	Pointer Arithmetic		
	5	Pointer Initialization	Deck.h	
	6	Comparing		
	7	Function Parameters	Almost every function	
	8	Memory Allocation	68	
	9	Return Parameters		
	10	Smart Pointers		
10		**Char Arrays and Strings**		
	1	Testing	164	
	2	Case Conversion		
	3	C-Strings		
	4	Library Functions		
	5	Conversion		
	6	Your own functions		
	7	Strings		
11		**Structured Data**		
	1	Abstract Data Types	Card.h, Deck.h	
	2	Data	Card.h, Deck.h	
	3	Access	66	
	4	Initialize	66	
	5	Arrays	68	
	6	Nested	66	
	7	Function Arguments	192	
	8	Function Return		
	9	Pointers	192	
	10	Unions ****		
	11	Enumeration		
12		**Binary Files**		

	1	File Operations	45-52	
	2	Formatting		
	3	Function Parameters		
	4	Error Testing	47	
	5	Member Functions		
	6	Multiple Files		
	7	Binary Files		
	8	Records with Structures		
	9	Random Access Files		
	10	Input/Output Simultaneous	46	

main.cpp

```
/*
 * File:   main.cpp
 * Author: Seth Tyler
 * Purpose: Project 1: Blackjack
 * Created on April 22, 2018
 */

// System libraries
#include <iostream>
#include <iomanip>
#include <string>
#include <cctype>
#include <ctime>
#include <cstdlib>
#include <string.h>
#include <sstream>
#include <vector>
#include <fstream>
using namespace std;

// Structures
#include "Card.h"
#include "Deck.h"

// Function prototypes
void intro();
void filDeck(Card*);
int getBet(int&, int);
void shuffle(Deck&);
void removeFromDeck(Deck&);
void deal(int amount, Deck&, Deck&);
void readable(Card);
void showSum(Deck&);
void addCard(Deck&, Deck&);
int bestSum(Deck);
void results(int, int, int&, int&);

int main() {

        srand(time(NULL));

        // Player's cash
        int cash = 100;

        fstream dataFile;
        dataFile.open("money.txt", ios::in | ios::out);
        if (dataFile.fail()) {
        cash = 100;
        } else {
        dataFile >> cash;
        }
```

```
dataFile.close();

// Repeat the game until player enters something other than Y or y.
char playAgain = 'Y';

// Run the intro
intro();

// Run the game
do {

char hit;

// Create the deck
Deck deck;
deck.size = 52;
deck.cards = new Card[deck.size];
filDeck(deck.cards);

int bet = 0;
cout << "Please place your bet. (Minimum bet: $6 | Current balance: $" << cash << ".00 : ";
cin >> bet;
getBet(bet, cash);
cout << endl << "You bet $" << bet << ".00\n"
        << "The dealer will now shuffle the deck and deal out the cards.\n";

cout << "Shuffling...\n";
shuffle(deck);

cout << "The dealer will now deal out the cards.\n";

// Create decks for dealer and player
Deck hand;
hand.size = 2;
hand.cards = new Card[hand.size];

Deck dealer;
dealer.size = 2;
dealer.cards = new Card[dealer.size];

int best = 0;

deal(2, hand, deck);
deal(2, dealer, deck);

// Display decks
cout << endl << "---------- DEALER'S CARDS ----------\n";
readable(dealer.cards[0]);
cout << "ONE CARD STILL FACE DOWN\n";
cout << "TOTAL SUM (SO FAR): " << dealer.cards[0].value << "\n";
cout << "------------------------------------\n";

cout << endl << "---------- YOUR CARDS ----------\n";
for (int i = 0; i < hand.size; i++) {
readable(hand.cards[i]);
}
showSum(hand);
```

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

        // Ask player if they want to stay or hit
        do {
        cout << endl << "Would you like to stay (keep your cards) or hit (request another card)? Press 1
to stay and 2 to hit: ";
        cin >> hit;

        if (hit == '2') {
                addCard(hand, deck);
                cout << endl << "---------- YOUR CARDS ----------\n";
                for (int i = 0; i < hand.size; i++) {
                readable(hand.cards[i]);
                }
                showSum(hand);
                cout << "--------------------------------\n";
        }

        best = bestSum(hand);
        } while (hit == '2' && best <= 21);

        // Check if the player busted
        bool bust = false;
        if (best > 21) {
        bust = true;
        cash -= bet;
        cout << endl << "BUST! You lose. Dealer wins.\n";
        cout << "You now have $" << cash << ".00" << endl;
        }

        // Play the dealer's turn
        if (!bust) {
        cout << endl << "Dealer's turn...\n";

        cout << endl << "---------- DEALER'S CARDS ----------\n";
        for (int i = 0; i < dealer.size; i++) {
                readable(dealer.cards[i]);
        }
        showSum(dealer);
        cout << "-----------------------------------\n";

        int dealerSum = bestSum(dealer);
        while (dealerSum <= 12) {
                addCard(dealer, deck);
                dealerSum = bestSum(dealer);
        }

        results(dealerSum, best, cash, bet);
        }

        // Ask to play again if money is sufficient
        if (cash >= 6) {
        cout << endl << "Would you like to play again? Enter Y for YES and N for NO: ";
        cin >> playAgain;
        }

        } while ((playAgain == 'Y' || playAgain == 'y') && (cash >= 6));
```

```
        cout << "Goodbye!";

        return 0;
}

void intro() {
        char tutorial = 'N';

        // Output the intro text
        cout << "Welcome to Blackjack!\n"
        << "This game uses one standard 52 card deck.\n"
        << "Would you like to learn how to play? Enter Y for YES and N for NO: ";
        cin >> tutorial;

        // Display the instructions if user inputs Y or y
        if (tutorial == 'Y' || tutorial == 'y') {
        cout << endl << "You will be dealt two cards. The objective of the game is to get the sum of your
cards to be as close to 21 as possible.\n"
                << "If the sum of your cards goes over 21, you will bust and lose your money.\n"
                << "In this game, it is you versus the dealer. If the sum of your cards is lower than the
dealer's, you lose.\n"
                << "If the sum of your cards is equal to the dealer's, the round will be over and you will
take back your bet.\n";
        }

        cout << endl << "Please note that in this game, there is no splitting, insurance, or doubling.\n";
        cout << "Let's get started.\n\n";
}

void filDeck(Card* deck) {
        int cardNum = 0;
        for (int suit = 1; suit <= 4; suit++) {

        // Create the standard 1-10 numeric cards
        for (int val = 2; val <= 10; val++) {
        Card newCard;
        newCard.suit = suit;
        newCard.value = val;
        newCard.type = 0;
        deck[cardNum] = newCard;
        cardNum++;
        }

        // Create the 3 face cards and an ace
        for (int type = 1; type <= 4; type++) {
        Card newCard;
        newCard.suit = suit;
        newCard.value = (type == 4 ? 11 : 10);
        newCard.type = type;
        deck[cardNum] = newCard;
        cardNum++;
        }
        }
}

int getBet(int &bet, int cash) {
```

```
        // Input validation for bet
        if (bet < 6 || bet > cash) {
        cout << "Your bet must be at least $6 and no more than $100. Enter a new bet: ";
        cin >> bet;
        cout << endl;
        getBet(bet, cash);
        }
        return bet;
}

void shuffle(Deck &crdDeck) {
        // Shuffle the deck
        for (int i = 0; i < crdDeck.size; i++) {
        int r = rand() % crdDeck.size;
        Card temp = crdDeck.cards[i];
        crdDeck.cards[i] = crdDeck.cards[r];
        crdDeck.cards[r] = temp;
        }
}

void removeFromDeck(Deck &crdDeck) {
        // Remove the first card from the deck and shift everything -1
        for (int i = 1; i <= crdDeck.size; i++) {
        crdDeck.cards[i - 1] = crdDeck.cards[i];
        }
        crdDeck.size -= 1;
}

void deal(int amount, Deck &crdDeck, Deck &mainDeck) {
        // Establish the first deck
        for (int i = 0; i < amount; i++) {
        crdDeck.cards[i] = mainDeck.cards[0];
        removeFromDeck(mainDeck);
        }
}

void readable(Card card) {
        // Output a card into a readable format for the player
        switch (card.type) {
        case 0: cout << card.value << " OF ";
        break;
        case 1: cout << "JACK OF ";
        break;
        case 2: cout << "QUEEN OF ";
        break;
        case 3: cout << "KING OF ";
        break;
        case 4: cout << "ACE OF ";
        break;
        default: "INVALID";
        break;
        }

        switch (card.suit) {
        case 1: cout << "HEARTS";
        break;
        case 2: cout << "CLUBS";
```

```
        break;
        case 3: cout << "SPADES";
        break;
        case 4: cout << "DIAMONDS";
        break;
        default: "INVALID";
        break;
        }

        cout << " (VALUE: ";

        if (card.value < 11) {
        cout << card.value;
        } else {
        cout << "1 / 11";
        }

        cout << ")" << endl;
}

int bestSum(Deck crdDeck) {
        // Calculate the best sum of a hand
        int sum = 0;
        int aceSum = 0;
        int numAces = 0;
        for (int i = 0; i < crdDeck.size; i++) {
        Card thisCard = crdDeck.cards[i];
        if (thisCard.type == 4) {
        sum += (numAces == 0 ? 11 : 1);
        aceSum += 1;
        numAces += 1;
        } else {
        sum += thisCard.value;
        aceSum += thisCard.value;
        }
        }

        if (sum <= 21 && aceSum <= 21) {
        return (sum > aceSum ? sum : aceSum);
        } else if (sum <= 21 && aceSum > 21) {
        return sum;
        } else if (sum > 21 && aceSum <= 21) {
        return aceSum;
        } else if (sum > 21 && aceSum > 21) {
        return (sum < aceSum ? sum : aceSum);
        }
}

void showSum(Deck &crdDeck) {
        // Output the total and best sum of a hand
        int sum = 0;
        int aceSum = 0;
        int numAces = 0;
        for (int i = 0; i < crdDeck.size; i++) {
        Card thisCard = crdDeck.cards[i];
        if (thisCard.type == 4) {
        sum += (numAces == 0 ? 11 : 1);
```

```
        aceSum += 1;
        numAces += 1;
        } else {
        sum += thisCard.value;
        aceSum += thisCard.value;
        }
        }

        if (numAces == 0) {
        cout << "TOTAL SUM: " << sum << endl;
        } else {
        cout << "TOTAL SUM: " << sum << " / " << aceSum << endl;
        }
        cout << "BEST SUM: " << bestSum(crdDeck) << endl;
}

void addCard(Deck &crdDeck, Deck &mainDeck) {
        // Append a card to a deck
        crdDeck.size += 1;
        crdDeck.cards[crdDeck.size - 1] = mainDeck.cards[0];
        removeFromDeck(mainDeck);
}

void results(int dealerSum, int best, int &cash, int &bet) {
        // Show results
        cout << "DEALER: " << dealerSum << "; YOU: " << best << endl;
        if (dealerSum > 21) {
        cout << "The dealer busted! You win!\n";
        cash += bet;
        cout << "You now have $" << cash << ".00" << endl;
        } else if (dealerSum == best) {
        cout << "Draw! You take your bet back.\n";
        } else if (dealerSum < best) {
        cout << "You have the better number. You win!\n";
        cash += bet;
        cout << "You now have $" << cash << ".00" << endl;
        } else if (dealerSum > best) {
        cout << "The dealer wins!\n";
        cash -= bet;
        cout << "You now have $" << cash << ".00" << endl;
        }
}
```

Card.h

```
/*
 * File:   main.cpp
 * Author: Seth Tyler
 * Purpose: Card structure
 * Created on April 22, 2018
 */

#ifndef CARD_H
#define CARD_H

#include <string>
using namespace std;

struct Card {
        int value;
        int type;
        int suit;
};

#endif /* CARD_H */
```

Deck.h

```
/*
 * File:   main.cpp
 * Author: Seth Tyler
 * Purpose: Deck structure
 * Created on April 22, 2018
 */

#ifndef DECK_H
#define DECK_H

#include <string>
#include "Card.h"
using namespace std;

struct Deck {
        int size = 52;
        Card* cards;
};

#endif /* DECK_H */
```