**Blackjack**

**Card Game**

Garret Koontz

CSC 5

Spring 2019

42409

**Number of lines: 361**

**Introduction**

Blackjack is a comparing card game between a player and dealer, meaning players compete against the dealer but not against other players. It is played with one or more decks of 52 cards. The objective of the game is to beat the dealer in one of the following ways:

* Get 21 points on the player's first two cards (called a "blackjack"), without a dealer blackjack;
* Reach a final score higher than the dealer without exceeding 21 or
* Let the dealer draw additional cards until his or her hand exceeds 21.

**Rules**

Single deck

Basic Play:

Hit: take another card from dealer

Stand: take no more cards AKA “stay”

**How to Play**

To start, the dealer gives each player 2 face up cards, and also gives themselves 2 cards. When all the cards are distributed, if the player gets an Ace and 10, they automatically have a Blackjack. The first non-Blackjack players choose to take a card(Hit) or no cards(Stand). If the player’s card number sum is more than 21, that player loses (Bust) and the dealer wins the bet regardless. If the player does not bust and decides to stand, then the turn moves to the next player. When all the players’ turns end, the dealer turns over his cards and continues to take the cards until the sum is not less than 17. If the dealer busts, the player wins the bet. Then compare the sum of each of the player’s cards.

**How I Came Up With the Code**

It took me about a day’s worth of planning to plan out how I was going to code a Blackjack game. I decided it would be easiest to create separate classes and headers to organize and initiate different parts of the game. Creating classes for the dealer, players, deck, and the game itself seemed the best option.

**Flowchart**

A close up of a map

Description automatically generated

**Checklist (list below is what is included) – could not put in line #’s due to time constraints..**

Cout

Integers

Characters

Strings

Floats

Bools

Comments

Type casting

Formatting output

If

If-else

Nesting

If-else-if

Logical operators

Validating user input

Conditional operators

Switch

Increment/decrement

While

Do-while

For loop

Input/output both

Function prototypes

Pass by value

Return

Returning bool

Static variables

Defaulted variables

Pass by reference

Overloading

Exit() function

1D arrays

Parallel arrays

1D as Function arguments

2D arrays

STL vectors

Passing arrays to and from functions

Passing vectors to and from functions