**Project 1**

Title

**Blackjack**

Course

**CSC 5**

Section

**42450**

Due Date

**April 25, 2014**

Author

**Victor Medel**

**Introduction**

Title: Blackjack

This is a simple program that allows any player to quickly play a game of Blackjack. The program starts up with a menu with three options, *1.* *Play Blackjack*, *2.* *Blackjack Game Overview,* or *Anything Else to Exit*. The options are very straight forward, option one allows the user to play the game, option two provide the quick overview of the game as written below, and any other input exits the user from program.

The object of the game is to beat the house by receiving a score of 21 or by getting a higher score than the house without going over 21 with any additional cards. The game begins by dealing two cards to the player; after displaying your score you will have the option to take another card to add to your total score or hold with your existing score. If you hold or go over 21 after choosing another card the program will automatically display the house’s hand and then determine the outcome. The card values are as follow:

Cards 2 through 10 = face value points

Jacks = 10 points

Queens = 10 Points

Kings = 10 Points

Aces = 1 or 11 depends on the player’s total. If player’s total is less than ten points then they

hold a value of eleven otherwise the card will hold a value of one.

**Summary**

Developing the program took about two weeks and several versions due my limited experience with C++ programing and the project packet development. As references I used the course textbook (*Problem Solving with C++ by Walter Savitch*) and the web to obtain some of the rules on how to play Blackjack as well as how to start this project. I also utilized the sample project documentation provided on Black Board to help me with the production of this document.

I’ve developed this program utilizing many of the concepts that have been covered by the class textbook (*Problem Solving with C++ 8th Edition by Walter Savitch*) within chapters one through four and partially five. I have also used concepts discussed during class lecture and lab to create this program. The program runs as expected but I believe that this program still has many opportunities. As I continue to cover new material in the class I believe I can improve this program and raise it to a more sophisticated programming level.

One of the major obstacles that I encountered while developing this program was the ability to pull an individual value from a function. Reading chapter five in the textbook I was able to find the **call-by-reference parameter**, which allows you to do just that. The **call-by-reference parameter** substitutes the function argument output for formal parameter of the function. This allowed me to pull the card value from the function making it easier to keep a running total score.

**Concepts Used**

From Textbook:

*Problem Solving with C++ 8th Edition by Walter Savitch*

Chapter 2

2.1 Variables and Assignments

2.2 Input and Output

2.3 Data Types and Expressions

2.4 Simple Flow Control

2.5 Program Style

Chapter 3

3.1 Using Boolean Expressions

3.2 Multiway Branches

3.3 More About C++ Loop Statements

3.4 Designing Loops

Chapter 4

4.1 Top-Down Design

4.2 Predefined Functions

4.3 Programmer-Defined Functions

4.4 Procedural Absraction

4.5 Scope And Local Variables

~~4.6 Overloading Function Names~~

Chapter 5

5.1 void Functions

5.2 Call-By-Reference Parameters

~~5.3 Using Procedural Abstraction~~

~~5.4 Testing and Debugging Functions~~

~~5.5 General Debugging Techniques~~

From Class Lectures and Lab:

1. Input and Output
2. Loops
3. Menus
4. Branching Constructs
5. Mathematical Expressions
6. User interactivity
7. Boolean Expressions
8. Functions

**Code Specifications**

|  |  |
| --- | --- |
| Lines of Code | 172 |
| Comment Lines | **29** |
| Blank Lines | **4** |
| Total Lines of Source Files | **205** |
| Number of Variables | **12** |

**Variables Used**

|  |  |  |
| --- | --- | --- |
| Type | Variable Name | Description |
| Integer | value | Function parameter that hold the dealt card value within the main program |
|  | suit | Function parameter that holds the suit of the card dealt within the main program |
|  | pTotal | Holds the players total score within the main program. It is also used as a function parameter that holds the players total score within the function definition and function header |
|  | rTotal | Utilized to keep a running total of the player score |
|  | total | Function parameter used to hold a card value total |
|  | hTot | Holds the house’s total score |
|  | hrTot | Utilized to keep a running total of the house’s score |
|  | choice | Menu selection input |
|  | crdValu | Function parameter that holds the card value within the function definition and function header |
|  | crdSuit | Function parameter that holds the card value within the function definition and function header |
|  |  |  |
| Character | ans | Input option to allow continue of play |
|  |  |  |
| Boolean | exitMnu | Alternative option to end program at menu selection |
|  |  |  |
|  |  |  |