Francis Hoang | CSC 5- 41202 | February 3, 2016

Project 1

Blackjack

# Introduction

This is essentially a C++ implementation of Blackjack since it seemed like something doable based off what we’ve learned so far this semester. Currently, at the time of this writing, it’s playable but it’s missing a dealer to play against, as well as money to bet with. It is simple Blackjack, currently with an infinite amount of decks so it’s theoretically possible (but highly unlikely) to get 21 Aces dealt. The player is dealt two cards, and from there they can keep drawing till they bust or choose to stand.

# Summary

This project mostly uses what we were taught, though I still have not thought up a good use for an int primitive type since Blackjack is a small numbers game, and short generally suffices for most calculations. I ended up implementing arrays since I wanted the player to know what their hand was, as well as eventually the dealer’s hand, so arrays were useful to store those values. The first big issue was handling the Ace since it could represent either a 1 or 11 that the player could change at will. I opted to have the player prompted every loop should they have an Ace, though it does interrupt gameflow a bit. In the process, I also realized I would have to reset the player point total to 0 and recalculate it again entirely instead of switching 1 and 11 to account for the unlikely scenario the player had multiple Aces which they may change as they please. I ended up repeating a lot of code due to this, which might be more smoothly handled as functions that I haven’t quite figured out how to implement just yet. How the game is ended to break the loop is somewhat inefficient as well at the moment, also a result of the recalculation of Ace since there was a period where the code would keep running even if the loss condition had already been reached. I had originally intended to use a for-loop for the drawing/hit me phase, since it would be a much smoother use of calling the values from the array than a separate counter, while the do-while was simply just to ask to play again, but I was somewhat haphazardly coding at the time so the do-while loop is the current draw phase for the moment. I may change it back to a for-loop in the future if I have time. Future plans are still to implement the dealer to add an additional win/loss condition for the player, as well as adding the gambling component of betting money. Other possible plans would be limiting the amount of decks, which would add the complication of having to account for a limited number of cards, and the ability to split your hand, which would involve having to account for another array.

# Summary v3

There was actually a V2 which basically a finished v1, and added gambling and a dealer as planned, as well as some polish, specifically in using the for-loop as originally intended. V3’s goal is mainly polishing it further by using functions to clean up the repetitive code. There’s still some code that’s repeated though, specifically the win-loss checks, which is mostly because I did not want to take the time to figure out how to make a function that would implement 4 or 5 other functions, where I would have to account for all those parameters for the sub-functions. There are two deal functions which probably could be consolidated into one as well, the initial 2-card deal, and the hit me deal. I mostly kept both deal functions since there was enough slight variation in the output that I didn’t want to combine the two. It does conveniently serves as an example of an overloaded function though. I have replaced the hasAce Boolean with a Boolean search function which basically checks for an Ace. I have also created a sort function which sorts the hand of both the dealer and player.