Project 1

Blackjack

Christian Villanueva
CSC 17C - 43324
6 June 2021

Introduction:

I have created a program that runs a simple version of the card game Blackjack from scratch. This version of Blackjack is the card aspect only. I have not yet implemented a chip system for betting, nor have I included the ability to split when a pair is drawn. Blackjack is my favorite card game, and it seemed fitting to make, given the requirements for the project, and my prerequisite knowledge of the game's rules. The entire program was created from the ground up, and no reference code was used until the final stage of the project, implementing trees, graphs, and heap sort.

The player is dealt two cards and one of the dealer cards is shown. The player is then given the choice to "hit" or "stay". If the player chooses to hit, then the player is given another card. If the player's card total (each card is given a number value, and is added up to find the total) hits or exceeds 21, they automatically stay. The dealer then reveals their second card. After the dealer reveals their second card, if their total is less than or equal to sixteen, they draw until their total exceeds sixteen. The card total of the dealer and the player are compared, and whichever total is closer to 21 wins, given that the total does not exceed 21 (if the player or dealer exceeds a total of 21, they lose the game). If the difference from 21 is the same or if both the dealer and player exceed 21 or "bust" then the game results in a draw. If the player chooses to stay, then their total remains, and the dealer continues the same as previously described.

Summary:

Total Lines: 1,678

Lines of Comments: 211

Number of Classes: 8

Number of Variables: 58

This project iteration took me about three days to complete, building off of my own Blackjack game from the past. Roughly 10 hours were put into the project including the documentation. The most challenging aspect of the project was implementing all of the trees and graphs into the program. I ultimately had to settle on replacing already existing containers and creating a distance problem at the end. Although I functionally understand how they work, implementing them into my project was difficult, and I ultimately had to come up with places to put them, despite the fact that the game is functional without them.

Github Link:

https://github.com/cv2808089/Villanueva_Christian_CIS_17/tree/master/CSC%2017C/Projects/Project%201

Description:

The game initially started as just a program that drew cards and output them. The card values and the ability to total them were then added, as well as the conditions for winning the game. Next came checks to ensure that repeat cards did not show up. After this came the rest of the concepts from the class up until this point such as pointers and binary files. After this came the splitting of the project into multiple source files, followed by conversion into classes. After converting into classes, I added exceptions, static variables and utilized a part of the STL. The next version added the copy constructor, and other minor tweaks. Following this was the addition of lists to the project. The next version added stacks, maps, and queues, which ultimately made the game more efficient, removing the need to check for repeat cards every time someone drew a card. The next concepts implemented were sets, and a sorting algorithm to find the min and max values played. After this came recursive sorts, where I replaced another sorting function I had with heap sort. The following version implemented AVL trees, where I took the existing program from the class github and the modifications made to it for homework, and used it to replace another container in the program.
The final version added graphs and hashing. Graphs were used to determine the minimum distance to travel home after leaving the casino. Hashing was used to check if an array was working correctly, by providing a means to check for a certain string quickly.

References:

AVL Tree:

https://github.com/ml1150258/2021_Spring_CSC_CIS_17c/tree/master/Class/AVLTreeTest

Heap Sort:

https://www.geeksforgeeks.org/heap-sort/

Graphs:

https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-using-set-in-stl/