George Velonis

2020-04-05

SNHU CS Capstone

Algorithms and Data Structures

This artifact is a piece of my final going back to CS-200. It is a simple program to input a roster of athletes for a sports team, then output them. It was originally made to show off my use of the basic list and dict data types in python, so it seems fitting that I used it to show off my use of advanced data structures and algorithms.

To demonstrate my ability to understand data structures, I created a player class to store data about the athletes including name and position, instead of the previous simple dict that only stored jersey number and rating. I then created generic classes for a doubly linked list and it’s nodes, then created a roster class to extend the DlinkedList class to this meet this specific application’s requirements. I then wrote functions to sort the roster by each piece of athlete data, mostly using bubble sort but one with insertion sort to show my ability to implement and understand different algorithms. Before starting the enhancements, I initialized the project with git to start using version control and linked it to my github so now I have a local and remote repository for it.

This artifact shows my ability to meet course outcomes CS-499-03 and CS-499-04. By implementing multiple algorithms and using object oriented design principles I have shown that I can apply appropriate computer science practice and standards and that I can use various techniques and tools to implement solutions that deliver real value.

I definitely learned a lot more about python when enhancing this artifact. I originally thought this aspect would be a relatively simple task of finding a reference implementation of a doubly linked list and sorting algorithms online somewhere and putting it to use in my application; instead I found a lot of challenges. Originally I implemented the player class as a type of node instead of just the payload for a node and ran in to struggles writing the swap\_node function. The sorting algorithms were also challenging because of the way I had structured my roster class. The bubble sort algorithms work in place on the roster, but I couldn’t figure out how to do that for the insertion sort algorithm. So instead it creates a new roster and returns it, which means when it is called from main it needs to have an assignment operator that the rest of the sorting funtion calls don’t need. If I had more time to put in to it I would try to refactor so that they all behave the same way, either operating on the roster in place or making a copy of it and returning the modified copy to then be assigned in main. I learned a lot about the python built in \_\_setattr\_\_ and \_\_iter\_\_ methods I needed to define when creating the classes. I also learned a lot about the self keyword when struggling with how to do insertion sort on the in place roster instead of creating a new one.