Andy Fleischer

Prof. Karla Fant

CS202

16 July 2021

Program 1 Efficiency Writeup

For program 1, we had to construct an event planner that could hold many different events of 3 types (I chose a gym event, restaurant event, and party event). For my class hierarchy, I decided to have an Address base class, a derived Event class for date and time, and from Event I derived Gym, Restaurant, and Party, which each had unique data and functions to work with. From Gym, I derived C_Node, a node with a next pointer to another C_Node, and a getter and setter. Similarly, I derived Node from Restaurant that does the same thing. The class CLL has a C_Node pointer "rear" and manages the C_Nodes as a circularly linked list through public functions and private recursive functions. The LLL_Array class has a Node pointer array "head" that manages Nodes as an array of linear linked lists. This approach was all proper single inheritance OOP, with the only exceptions being the node-type classes that needed getters and setters. Overall, I believe this approach was very effective and valid.

Going through the process, I did not ever need to make drastic changes since I did a lot of planning before implementing (measure twice cut once). The biggest changes came from shifting from implementing one event of each type to implementing many through a data structure. This posed many challenges, like how to access the functions of Gyms, Restaurants, and Parties through a data structure with many of them. For this, I ended up creating "find" functions that found a certain instance the user wanted of the event type, allowing them to call functions on these events.

In terms of efficiency, there are some pros and cons to both the array of LLLs and the CLL. Both are actually fairly fast at insertion, since the CLL always inserts at rear->next, meaning it takes constant time, and the array of LLL inserts at a random index at the head of the list. However, because the LLL array inserts at random indices instead of, say, hashing to an index, searching for a specific restaurant is highly inefficient. The CLL also requires will traversal and is thus inefficient as well. I think that the best thing I can do for the LLL array is implement some

sort of hash function, probably in the Node class, to allow for faster direct searching. This will lose the ability to sort data, however, so another improvement I could make would be to sort my data somehow, probably by date. If I did this, I might also use upcasting and dynamic binding to create one large data structure that can hold any event type in it for much easier code.