Chase Verbout
CS202 Spring 2022
Program 3

Efficiency Write-Up

Prelude: This program assignment was my first true experience with going full steam on a large implementation only to come to a much greater understanding half-way through. I forced myself to continue forward but really, I just wanted to scrap it all and try out my *new, amazing, better* ideas. Seeing as this is the reality sometimes; we get far along in a real-world project and re-starting is not always a feasible option, we must instead work with what we've got to get a similar result. All that to say, this code is not my favorite work that I've produced, but it is definitely interesting.

1. **Discuss the effectiveness of your design and classes**
    1. A working program that fulfills its promise of containing various object types in one data structure. So, in the vein of functionality, this program effectively gets the job done. I started it out right with an Abstract Base Class Concept and 3 derivatives: STL, Modern, Python. From here I was able to correctly implement my pure virtual classes. For my data structure: As I understood the outline, I was to make a BST that contained nodes with LLLs of objects with the same website. The direct reference to LLL led me to believe I had to implement it rather than utilize an STL data structure. Thinking I was being clever, I started with a base Node Class that contained LLL functionality (get next, set next, has next, ect) and derived a BST Node class from this with left and right functionality. This way every BST node has a possible LLL but the nodes in the LLL do not have the BST functionality. It was amazing.
    2. Then I had to implement the BST. Making a BST work while simultaneously supporting the possibility of an LLL was exhaustive. Truthfully, in a perfect world where there are no bugs in my BST LLL combo, it is probably more efficient than the STL list. It is less complex than an STL list because it has only the functionality that it specifically needs and, in that sense, I could see it being less consuming than the STL. It is a data structure specifically made to be used with the data of a base class pointer and so it may be that it is quicker overall with what the program does.
2. **Discuss the validity of your approach**
    1. With my previously stated understanding of the assignment- I think that my approach was valid. However, knowing that an STL could have been used as the LLL I would say that this approach is much less valid and relatively outlandish. Two data structures should not be worked into the same one. With a lack of more time though I did opt to not make two separate data structures (BST and LLL) .

Regardless of this, using an STL would have made my code considerably less complex and more readable. It was devastating to find out because getting this to work out as it has was difficult but yet still satisfying to complete. Validity in the case of this program requires that we have a much more specific outline and that we ultimately all interpret it similarly. Without strict rules validity can usually be argued one way or another.

GDB Write-Up

1. **Discuss the effectiveness of the debugger**
    1. This week I didn't have as many problems that required the use of GDB. I think it had to do with the emphasis of the program. In past weeks we've been working on features that convolute the programming process a little further. But this week with dynamic binding the changes are all really in the core hierarchy and a small portion of the calling functions in the data structure(RTTI). But getting the core hierarchy to work was not as difficult as following some data through a unique_ptr in a template.
2. **Discuss what problems it helped you solve**
    1. A specific example of what GDB helped me figure out this week was in verifying that the downcasting was calling the correct class derivatives function. I wouldn't say that this was solving an issue but more so that I was making sure that things were working as I was told they would and wanted them to.

       I did something similar again with my BST wherein I made sure that it was correctly swapping data between nodes. This was great with the removal algorithm- I set up the nodes with a swap method so copying any data was unnecessary and I could delete the chosen leaf node that had been given the swapped value. Extremely nice for the IOS process.