**Ernesto Martinez**

**CS 202**

**Program 3 Analysis and UNIX debugger**

**Feb 27, 2017**

Analysis of Program

The data structures performed excellent for this program. As the assignment requested, I implemented a Binary Search Tree in which every node contained a Linear Linked List of First names corresponding to the last name stored in each node of the tree.The tree class managed all the operations corresponding to the insertion, displaying, and removing from the list. The insertion function was inserting the last names in sorted order so the display function had an easier job displaying them all. The removing function was assisting the destructor to deallocate all the memory allocated by the application. It will be the last time in this course using a destructor since Java has it's own garbage collection. The destructor of both classes managing the data structures worked perfectly since there were not memory leaks in the program. The linear linked list class managed the operations vinculated to the insertion in order to the list, the displaying and the removing nodes. All the functions were tested step by step and worked fine. I had to use the debugger for some parts during the sort insertion.

I do not think that another data structure would've worked better, maybe a linked list of arrays but would not make so much difference since the bst was more appropriate due to the easiness of the sorted insertion.

The data structures part of my design was efficiently and the way it communicates to a manager class that control the program performed very well, just as my design described.

Very few things of my design were not efficient. I had to change some algorithm to approach the solution based on the operator overloading part. Those adt classes had some misconnection at the beginning but I changed them later on the implementation part in order to make it work.

If I had more time I would spend it a little bit more in the operator overloading logic since at the beginning it looked trivial to me since is the first time I use it.

The classes had clear responsibilities, as I explained before when I described the data structures. The manager class took the control of the application and managed operations such as welcoming the user, prompting and also have a data member which is an instance of the bst class to manage those operations related to the data structure, without having to the declare the object in Main.

No classes did the job that other classes should do. I spent some quality time in the design to make sure that the classes managed their own members and responsibilities. The hierarchical relationship were in the classes which used operator overloading. An emergency is a broadcast system and more. From my point of view, an emergency inform people by starting an alert call but also besides the responsibility of a broadcast system, it tries to manage the situation and end the emergency ( hurricane, accident etc.) by taking action. Operator overloading was used here as part of the assignment implementing assignment, equality and output operations.

I tried to make the program fully object oriented but maybe in some cases my algorithm went in a different way like in the insertion functions, where I used head->next instead of a function call to do it for me.

Debbuger

The debugger was very efficient when dealing with the data structure specially since I used recursion in all my functions. It was key in the insertion sort and also important to identify some bugs with the hierarchies. It also was efficient when I tested my operator overloading functions to check if they were working well since the compiler could not verify that for me.

Besides the debugger, I used valgrind a lot. At the beginning I encountered a problem that i never saw before, I was

deallocating more memory than the memory I allocated. Valgrind helped me to verify where exactly I was making this mistake, more precisely, using Valgrind --leak-check=yes. By far, the biggest help in my program, since it helped me solve the leaks and finish with all being deallocated well.

The debugger is helpful to enhance my programming experience because I can follow my program line by line and using the print tool to verify the data inside a variable and was useful to set breakpoint when I wanted to skip some part of the program.

I would like to know features in the debugger that could help me with inheritance. I do not know if features like that exist but I would like a tool that show a map diagram of how the program is working.