Design #4
Author: Alves Silva, Otavio Augusto
Date: 05/30/2015
CS 163 - Program #4

1) How well did the data structure selected perform for the assigned application?

The table(Binary Search Tree) selected for this assignment was able to achieve the objective that was assigned. The tree is totally capable to work with an unknown upfront memory that the system would need in the future. Also, the table was capable to work with an ordered and sorted data. Therefore, the data structure proved a fast away to access the data even with an unknown upfront memory that would be necessary in the future.

2) Would a different data structure work better? If so, which one and why?

Yes, We could change the kind of the tree. We used a sorted binary tree, so the tree isn't balanced in the compilation time. If we have used a 2-3 tree or a 2-3-4 instead the tree would be balanced all the time. However, the usage of this tree is complicated because of the remove algorithm is a little bit hard to implement. Therefore, a different kind of tree would be a good choice if you already know the implementation of the remove algorithm that will be used by the program.

3) What was efficient about your design and use of the data structure?

The tree was capable to achieve all requirements and functions to be considered efficient. The retrieve information was able to find the applications with a specific. The remove by keyword was capable to remove any node in any position as a leaf, with one child or two children. The data structure proved a good choice whereas we don't know the amount that will be necessary with a fast access to the data.

4) What was not efficient?

The insert function inserted one node for each keyword in the app. So, we have one app that would be pointed by one or more nodes. Therefore, in the moment that we desire to remove an app by a specific keyword, but it has more than one keyword. We will remove just that specific keyword, the others will continue pointing to the application. Another issue that the insert function produces is that the display function can display more than one time the same app.

5) What would you do differently if you had more time to solve the problem?

I would change my away to insert the app to resolve the issue with the repeated applications. Therefore, the remove and the display would work properly in the system.

6) The differences between the BST and Hash tables

- The hash table needs a specific size before starts the program. The BST can grow by the compilation time.
- If the size of the hash table is small probably we will have more collisions and the chaining of the table will be bigger. The BST will not have collision because we are using a sorted data structures.

- Improving the size of the hash table by two can do a visually change in the shape and the collisions of the hash table. However, the empty spaces of the hash table can be found more easily. Therefore, the wasting of memory is occurring. We don't waste memory using the BST, since it grows in compilation time.
- Using a sorted data structure like the BST, the speed to find a specific item is less than using the hash table.
- The recursion works perfectly with BST. However, with the hash table it could be a problem.