Design #1
Author: Alves Silva, Otavio Augusto
Date: 04/19/2015
CS 163 - Program #1

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

The data structure selected for this assignment was able to achieve a part of the objective that was assigned. One of these has the following command: organizes a list of a furniture parts by reading a .txt file by the number part. Therefore, the code was able to organize a list in ascending order from a totally disorganized text regardless of the order of the parties concerned as well as it was capable to find and see a single part in the list to see the characteristics (Part number, description, amount, steps) of it. The ADT complete the most important parts of this assignment, which were the creation of classes and ADT to manage the data in a organized away.

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

I created a class Part for this assignment, but a struct could be better for me. However, I didn't choose this one because I had done some exercises using a class, so the implementation was clearer to me. The use of a struct would be easier to create the code because structure has less complexity than a new ADT with a big amount of variables. Since, the members are already public and we don't need to create a function to retrieve the data. Therefore the assignment could have been more easily implemented using a struct than a class for the part.

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

My ADT has all the minimum requirements and design to be considered efficient. It is able to be replaced without affecting the client program directly and has the operations for this, which are insert, display, find, copy information and etc. The algorithm to order the parts in the list is totally able to work in different away like: If the list is empty, if the next part has to be in the first position, if the next part has to be in the final of the list or in the last position. Therefore, the codes even as the algorithm were able to achieve the objective.

4) What was not efficient?

I could not implement a list by step. I didn't find an away to do it. Therefore the code is totally inefficient to do it. Another think that I realized now is that I could use a tail pointer to help me add at the final of the list without going until the last node to see that the node has to be there.

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

I would change my away that I implemented my part data structure. Now I see that the implementation of a structure could have facilitated the resolution of the problem and spared long enough for me to try to find a way to implement the phase of a list by steps.