Systems Programming Assignment 2

Mariam Tsilosani and David Awad

The following are the structs we decided to use.

```
struct SortedList{
    CompareFuncT cf;
    DestructFuncT df;
    struct Node_ *head;
};

struct Node_{
    void *data;
    struct Node_ *next;
    int reMoved;
    int refCount;

};
```

- Implementation :

We decided to use a Linked-List to solve this programming assignment. We stored the nodes inside a sortedList struct along with the comparator and destructor functions. We use the comparator function to determine the placement of the new object node when SLInsert is called. The refCount is an integer keeping track of the references to each specific node so that helps us know not to delete the node when the iterator is pointing to it.

- Time Complexity :

Our implementation runs in O(n) (n being the number of nodes currently in the list). Since the worst case for insertion would be placing the node at the end of the list (traversing through the whole list) and same for removal (Worst case would be O(n)).

- Space Complexity:

We allocate memory only to create new nodes containing the hidden data of unknown size. The size of each struct is completely dependent on the size of the arguments to the program which is why it must be dynamic.