**CS 2302 Data Structures**

**Spring 2019**

**Lab Report #3**

Due: October 4th, 2019

Professor: Olac Fuentes

TA: Anindita Nath

Author: Kevin Ramirez

**Introduction**

For Lab #3 we had to implement a Sorted Linked List, a linked list with all of its elements being sorted in ascending order, with multiple other method tools, and these are “Print”, a simple method that prints all the elements of the list in ascending order, “Insert”, this method will add a new node with new data and then sort the list again. “Delete”, this deletes a specific node, so long as you know the correct index position of the element that you want to delete. “Merge”, you give it another Linked List of the Sorted Link List type and it will be added or merge to the original Link List from which you call them method, and the list will continue to stay sorted. “IndexOf”, just returns the index of the element that you ask for. “Clear”, will erase the link of the head of the list and set both the head and the tail to the same None element. “Min”, if the list is sorted, it should return the head which must be the smallest element. “Max”, Same as Min, if the list is sorted, then return the tail. “HasDuplicate”, returns true if there are duplicated data values in the list. “Select”, returns the k-th smallest element in the list given a value for k.

**Proposed Solution Design and Implementation**

**Operation #1 Print:**

Print it is a simple method that uses a while loop to iterate through the list and then print the data in each node, if the head of the list is empty it will print “list is empty”.

**Operation #2 Insert:**

For the operation Insert, we utilize two other already existing methods, “addNode” which takes an integer value and adds it to a new node that gets added to the end of the List, and “sort”, this method does not change the pointers of the list but it sorts the data with in each node, so it the first node has a greater value than the second one, it will hold the data value of the second node, then it set the data of the second node to the first node, now both have the same data value, but at the end of the while loop it will set the hold data value to the first node, so now the first node has the lesser value and the second node has the greater data value.

**Operation #3 Delete:**

“deleteNode” will take a given integer value, then with a for loop it will go to the node previous to the node that we want to delete, then skip over the node that we don’t want and set it to the next one. Else if the node that we don’t want is the head, just set the head to head.next. If the given number is higher than then the size of the list it will not do anything.

**Operation #4 Merge:**

For the “Merge” method we call it from the List to which we want to merge the given list to, so if you want to add a list to you first list you would type list.merge(new\_list), the given list will be turned into a normal list, then that list will be iterated with a for loop and using the method “addNode”, we will add to the original list, then we just call the method “sort”, to sort our new list.

**Operation #5 IndexOf:**

The method “indexOf”, will travers the list a counter, when the counter equals the same as the “i” index it will return the appropriate data value in the sorted linked list.

**Operation #6 “clear”:**

The “clear” method will simply set the head and the tail to None, to the list from which it was called, there is no need to do anything else since python will take care of the left over data and delete it.

**Operation #7 “Min”:**

The way I implemented “Min” method, was to simply return the head of the list, since we have a method that sorts the list for us, all we have to do is get the head and return it, since the head of the list must be the smallest element in the entire list.

**Operation #8 “Max”:**

The “Max” method does the same thing as the method “Min”, but instead of returning the head, I will return the “tail”, because if the sort method does the correct thing it should set the greatest element in the list to the very tail of the list.

**Operation #9 “HasDuplicate”:**

The Method “hasDuplicate” will use two iterators and two while loops, the first while loop will hold an element of the list and the second embed while loop will iterate from the rest of the list until it finds a duplication of the value being hold and return True. If there are no duplications the it will return False.

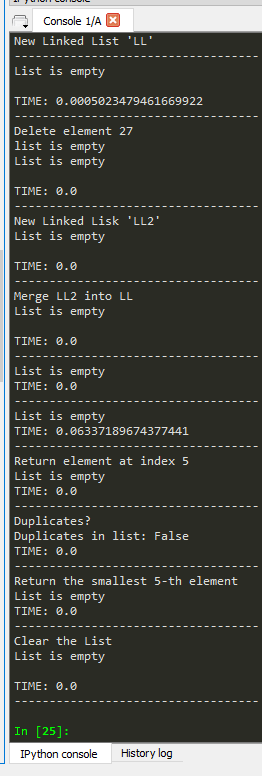
**Operation #10 “Select”:**

For this method, we use the same logic as in the method IndexOf, but here instead here we will use a for loop and iterate through the list until we have found the index of the element that we want, then it will return that element which is the smallest k-th element. Otherwise it will return math.inf.

**Experimental Results**

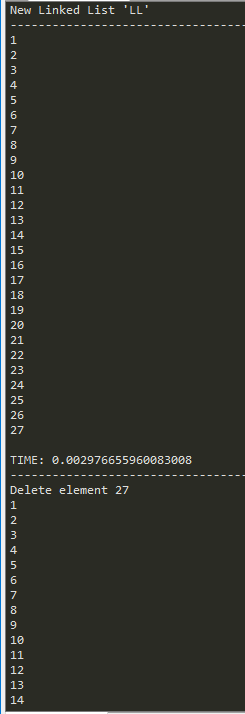
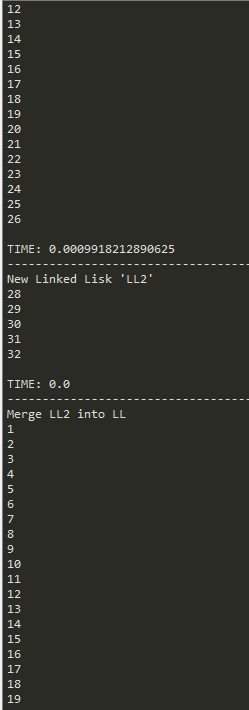
The test cases for the all the operation are just the same elements, since what matter is whether or not the methods work properly, so we will have only two cases, the list is given multiple values or none. Empty list or a full list.

**Case #1 List is Empty for all:**

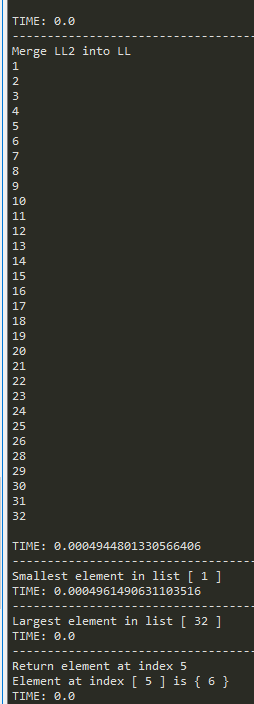
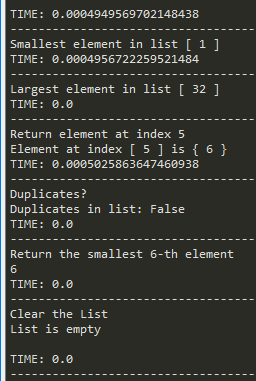
****

**Case #2 List is Not Empty for all:**

**[ 1 ] [ 2 ]**

** **

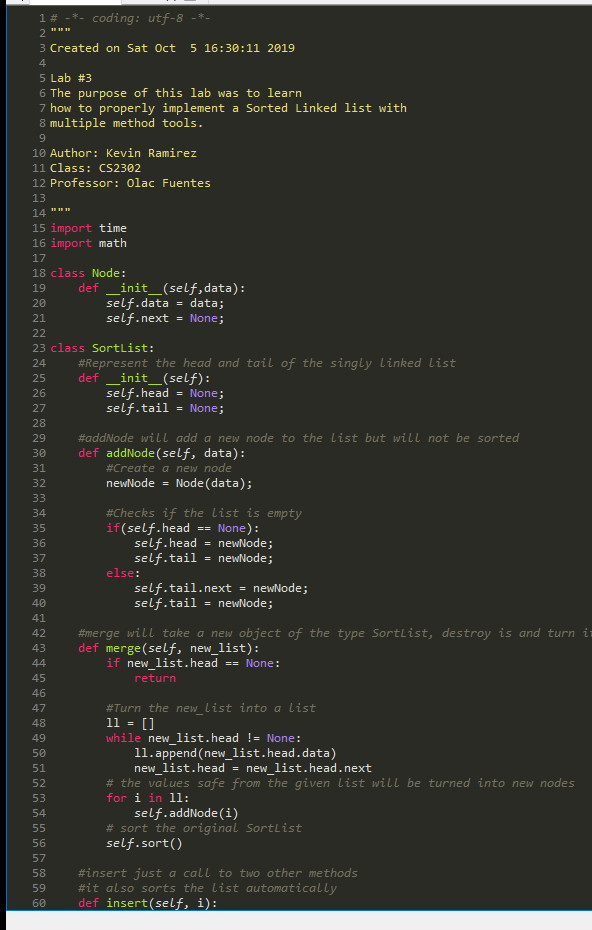
**[ 3 ] [ 4 ]**

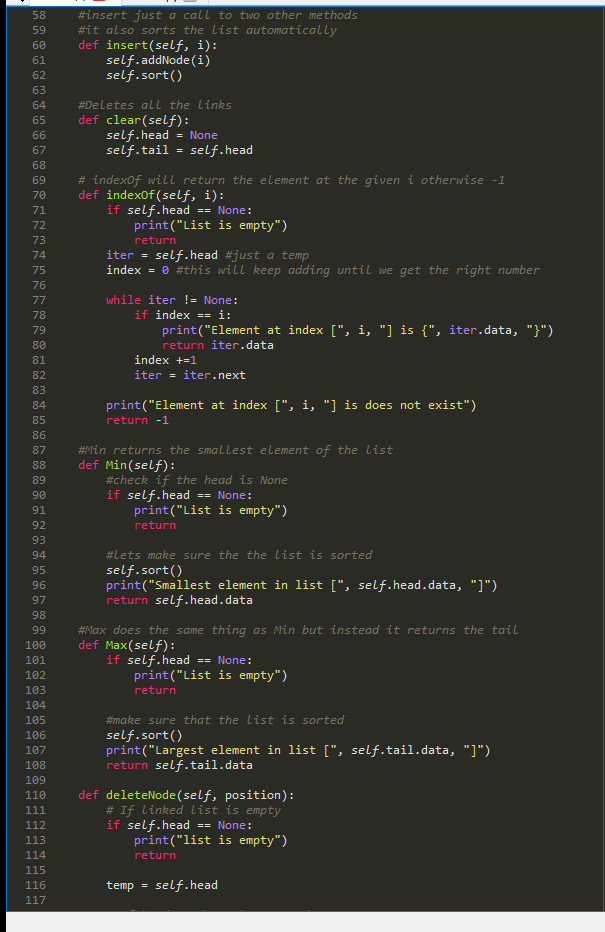
** **

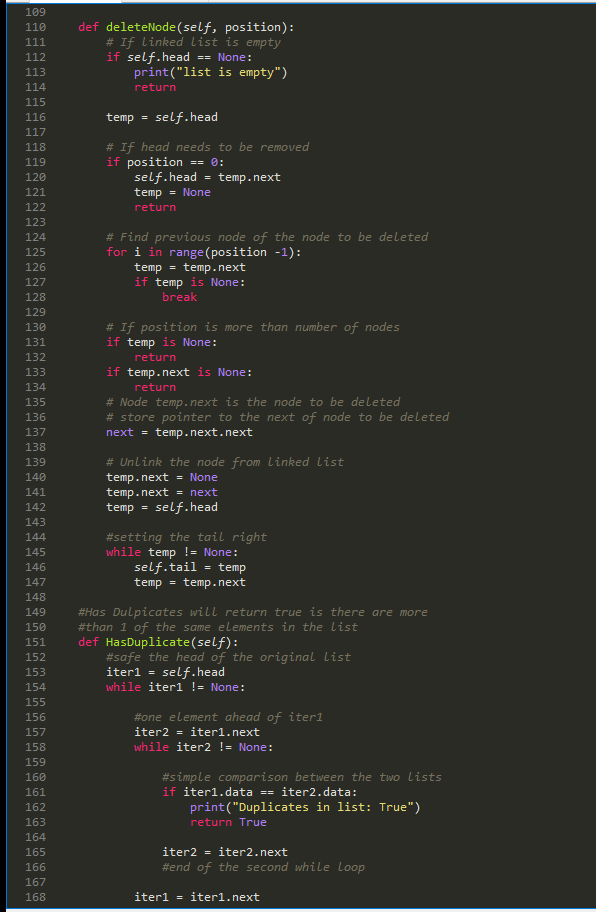
**Conclusion**

In conclusion, this lab was easy and fun, some of the methods looked complicated but to my surprise they were pretty simple to implement. I learned how to implement sorted linked list in python and realized that doing this is way easier with python then with java. This helped to reinforce my understanding on how linked list work.

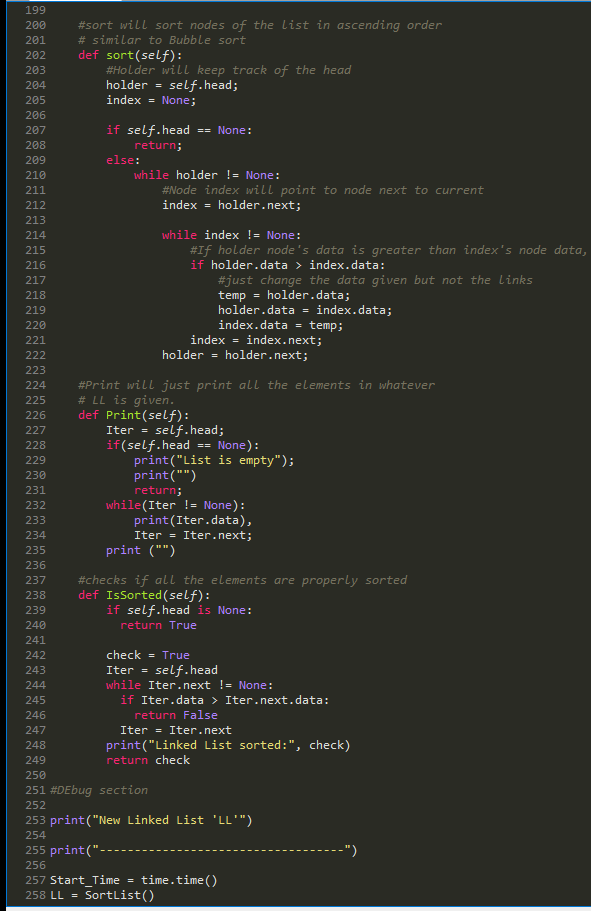
**Appendix**

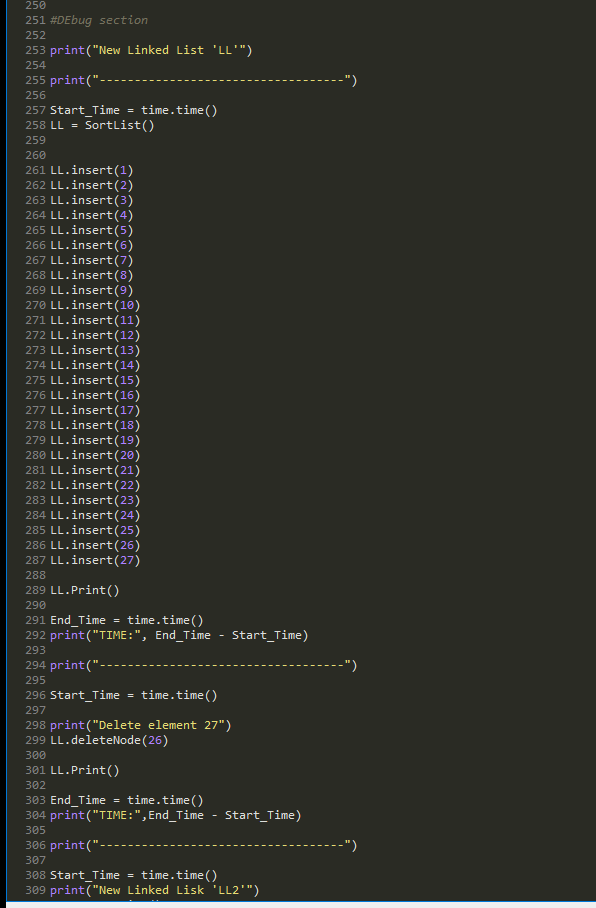
****

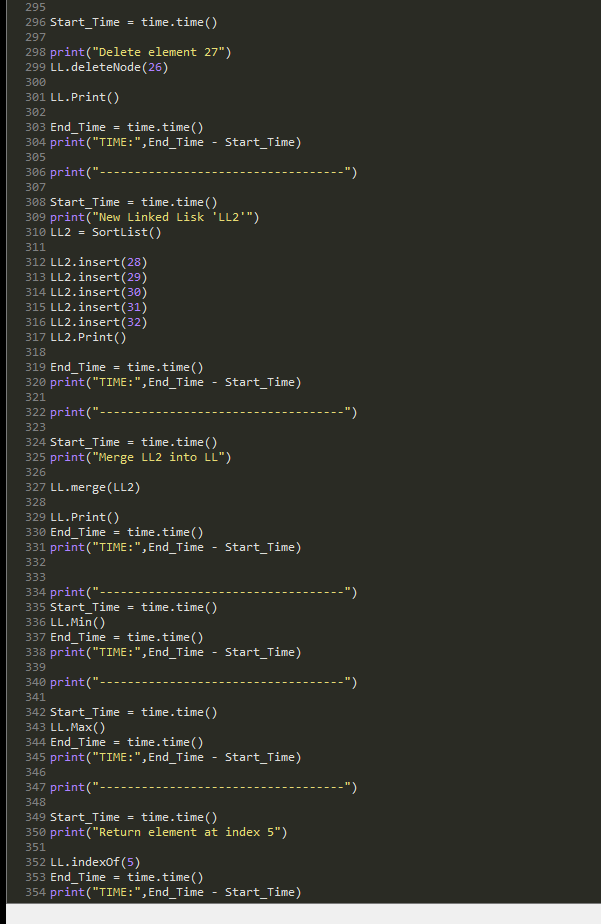
****

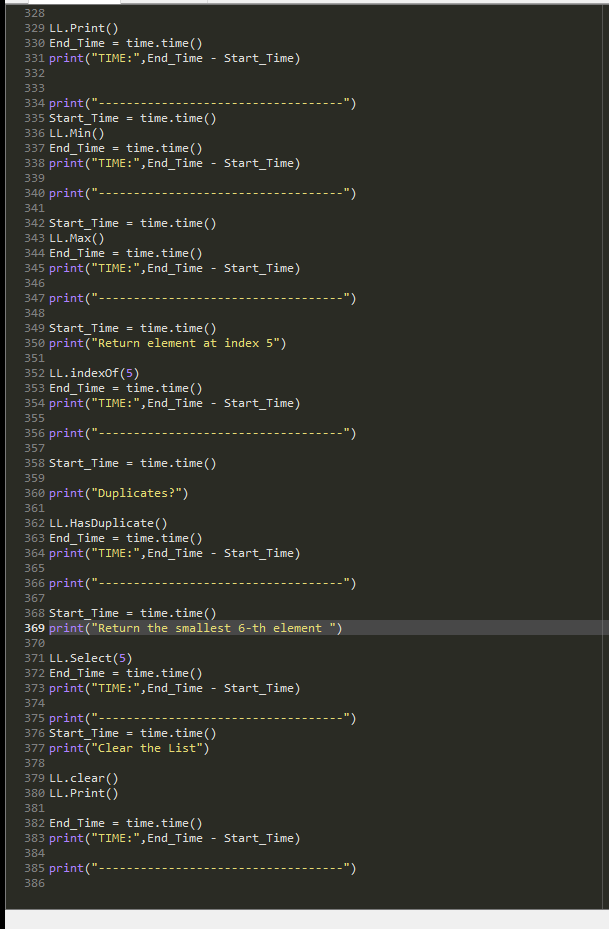
****

****

****

****

****

****

I certify that this project is entirely my own work. I wrote, debugged, and tested the code being presented, performed the experiments, and wrote the report. I also certify that I did not share my code or report or provided inappropriate assistance to any student in the class