**CS 2302 Data Structures**

**Fall 2019**

**Lab Report #3**

Due: 10/03/2019

Professor: Olac Fuentes

TA: Anindita Nath

Programmer: Miriam Olague

**Introduction**

For this lab, we were asked to work with linked lists. We were asked to print the entire list, insert a new node, delete a node, merge two lists of type SortedList, find the index of an integer, clear the entire list, find the minimum value, find the maximum value, check if a value has a duplicate, and returns the kth smallest element. The main objective of this lab is to learn how to modify linked lists different ways.

**Proposed Solution Design and Implementation**

**Part #1 Print(self):**

For this operation, I used the Print() method on QUIZ\_3 on replit. I moved from node to node and printed it.

**Part #2 Insert(self,i):**

For this operation, I used Ch 4.2 of Zybooks as a guidance. I made sure to understand the process of the code so I could modify it and be able to insert a new node. I saved the head into a temporary value and the tail as well. Since the list is a sorted list, I was checking if the new node that wanted to be inserted was in between two nodes, the left node had to be smaller than the new node and the right node had to be greater than the new node. I then inserted the node where it belonged.

**Part #3 Delete(self,i):**

For this operation, I also used Ch 4.2 of Zybooks as a reference. I just identified the number that wanted to be deleted and kept a reference point to the previous node of that node and pointed to the next node.

**Part#4 Merge(self,M):**

For this operation, I sent the head of self and the head of M as parameters. I checked if the data of self was smaller or equal to the data of M and if It was, I made a recursive call in which I moved to the next node of self. If M’s data was smaller, I made a recursive call in which I moved to the next node of M. I then returned the list.

**Part #5 IndexOf(self,i):**

For this operation, I used the code of ‘ElementAt’ I wrote for QUIZ\_3 on replit as a reference. I modified it so it could move indexes each time I moved to a next node.

**Part #6 Clear(self,i):**

For this operation, I just made the head be None so it could clear the list.

**Part #7 Min(self):**

For this operation, since it’s a sorted list, the first one must be the smallest one. I returned head.

**Part #8 Max(self):**

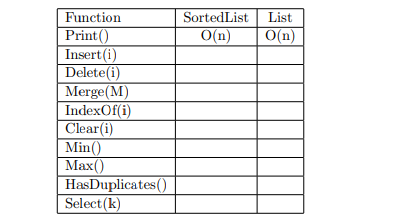
For this operation, since it’s a sorted list, the last one must be the biggest one. I returned tail.

**Part #9 HasDuplicates(self):**

For this operation, since it’s a sorted list, I checked a node and the next node to see if the elements were equal, if they were, I returned True. If not, I kept moving to the next node.

**Part #10 Select(self,k):**

For this operation, since we had already done this for Lab #2, I kind of knew what I had to do. I used Part #5 IndexOf(self,i) as a reference. I iterated through the list and returned the value of the node.

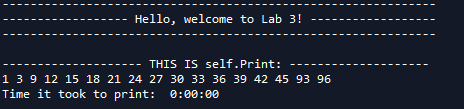
For the chart, a sorted list is more efficient rather than a normal list since we don’t know where we will find the element that is being looked for.



**Experimental Results**

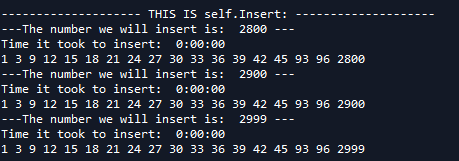
**Part #1 Print(self):**

For this operation, I did not have any trouble at all since I had this code in Java and the code was given to us on replit. It helped me understand and remember how linked lists work.



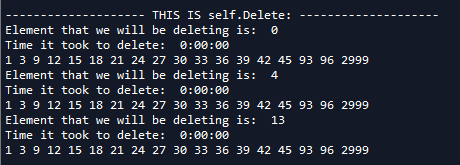
**Part #2 Insert(self,i):**

For this operation, I had a hard time coming up with something that would let me insert the node where its supposed to go. It was a lot of trial and error but then I created a temporary variable that kept track of head and another one that kept track of head.next so I could check the value of both and check if the node belonged in between of them.



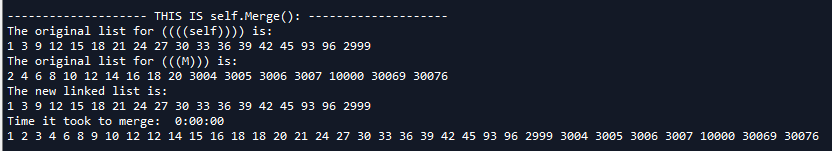
**Part #3 Delete(self,i):**

For this operation, I approached it the same way as I approached Part #2 Insert(self,i). I created a temporary variable that kept track of the head and another one that kept track of head.next so I could check the value of both and check if the node belonged in between of them. I then erased the node.



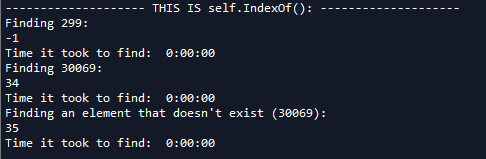
**Part#4 Merge(self,M):**

For this operation, I tried to make this function inside of a class but it wouldn’t let me do it. I then decided to do this function outside of the class. I first thought it would be a good idea for me to do this non-recursively but then I started to run into problems. I did not know how to append the items to the list one by one. I then decided to do it recursively and append items in the right order.



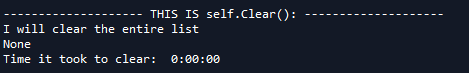
**Part #5 IndexOf(self,i):**

For this operation, I got it right the first time. I kept a counter and compared it to the element that the user is looking for and if my counter equaled the user’s input, I returned it.



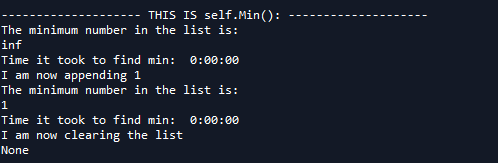
**Part #6 Clear(self,i):**

For this operation, I got it right the first time. I just cleared head.



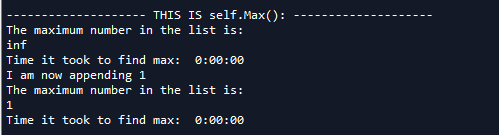
**Part #7 Min(self):**

For this operation, I got it right the first time, I just returned head.



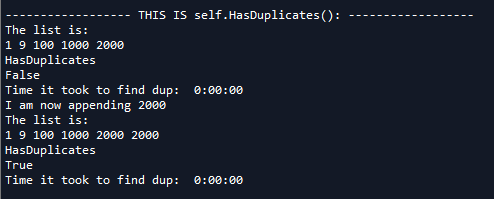
**Part #8 Max(self):**

For this operation, I got it right the first time, I just returned tail.



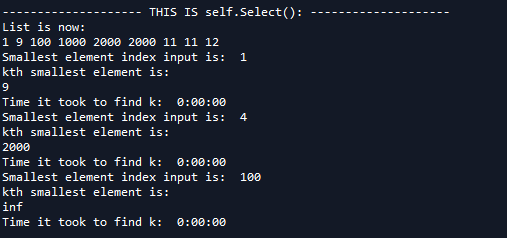
**Part #9 HasDuplicates(self):**

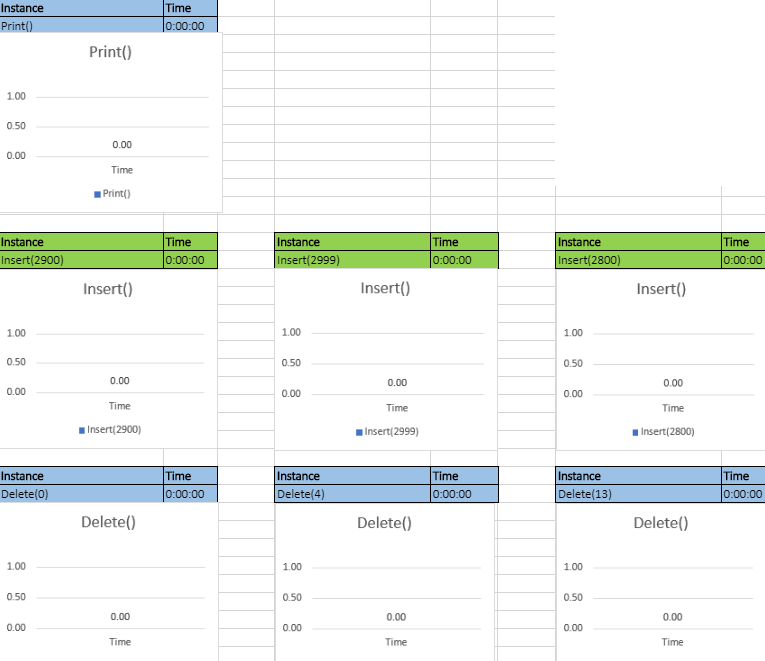
For this operation, I had a hard time coming up with an answer. I tried moving from node to node and saving it into a temporary value so I could compare it to the rest of the nodes but then remembered that the list was sorted, so the next value would contain the same value.

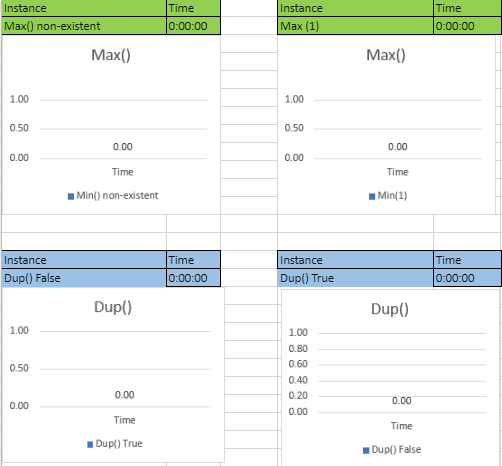
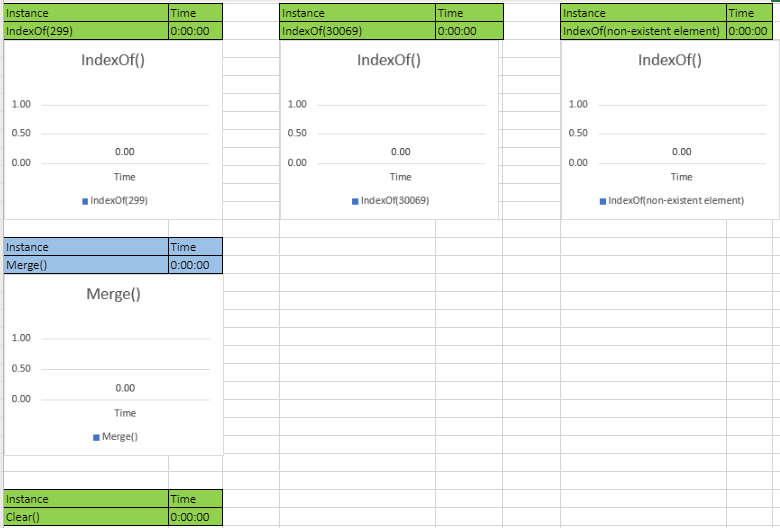


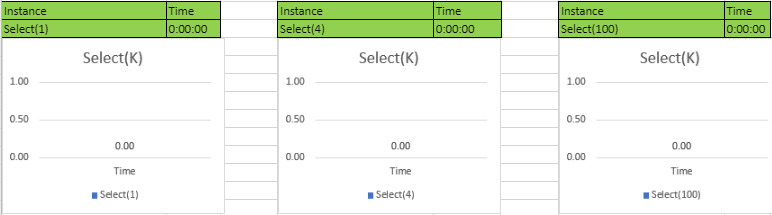
**Part #10 Select(self,k):**

For this operation, I got it right the first time since I just needed to find the kth element and return it.







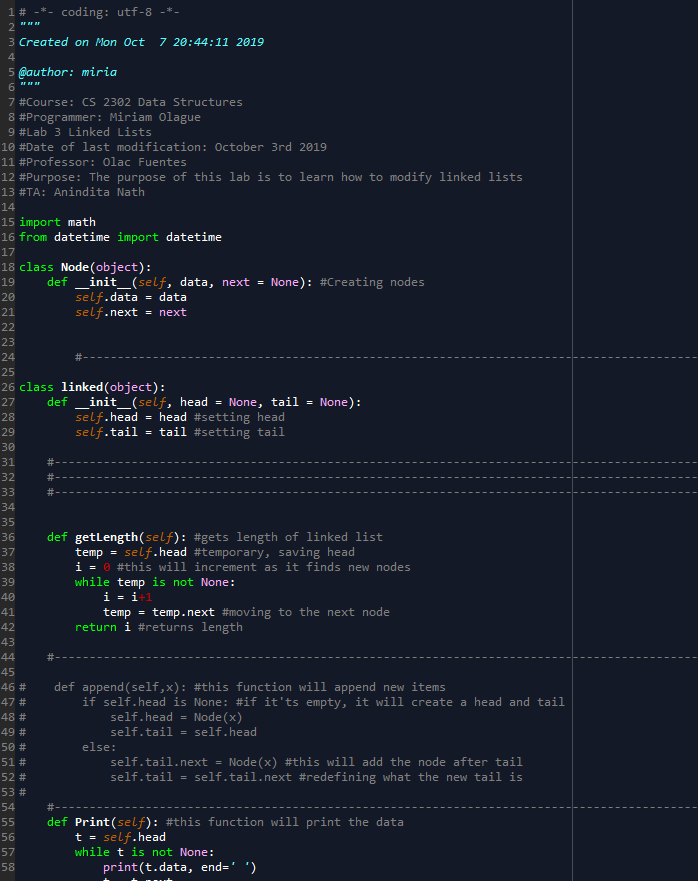


As the results show, I had a hard time making the times change. I appended up to 60 elements and the time it took to call the method and return the output desired, but the time still remained as zero. Every single method I called did took 0:00:00. I attached different graphs for every function I created so the times don’t overlap with each other.

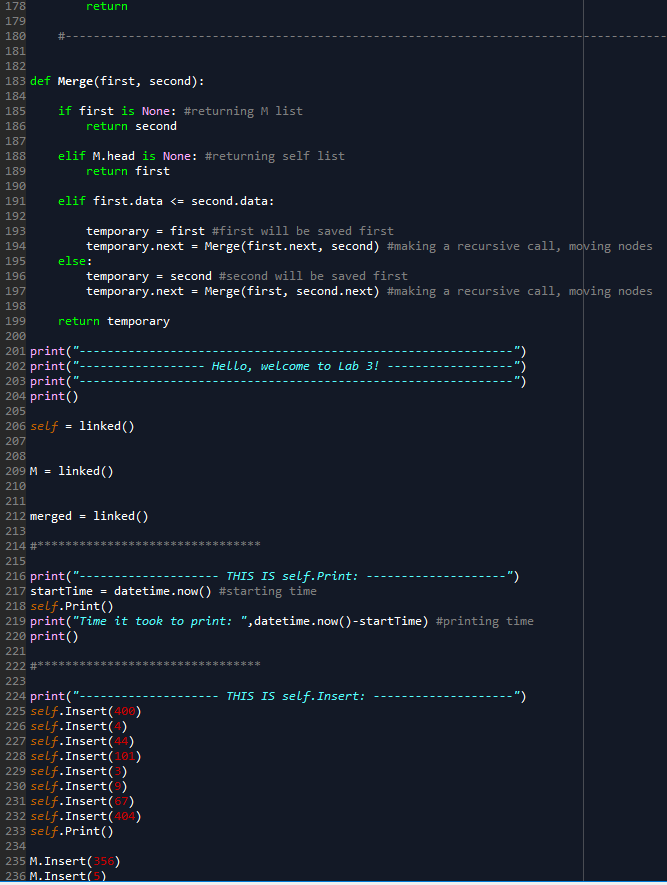
**Conclusion**

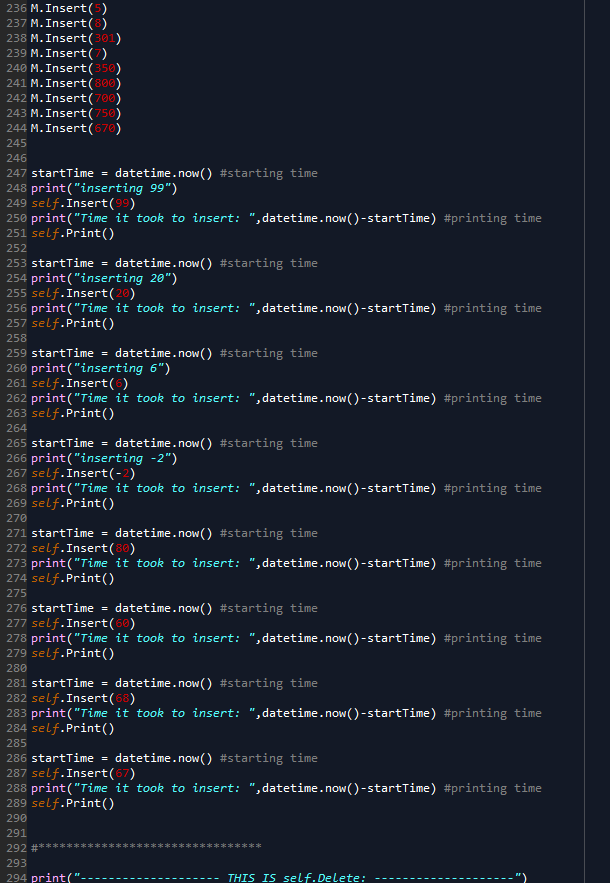
In conclusion, this lab was very easy and helped me remember everything I knew about linked lists since I saw this in CS2. It was not very difficult to solve since it was a very easy lab. I feel like I have mastered linked lists now and will be able to solve almost any problem given to me.

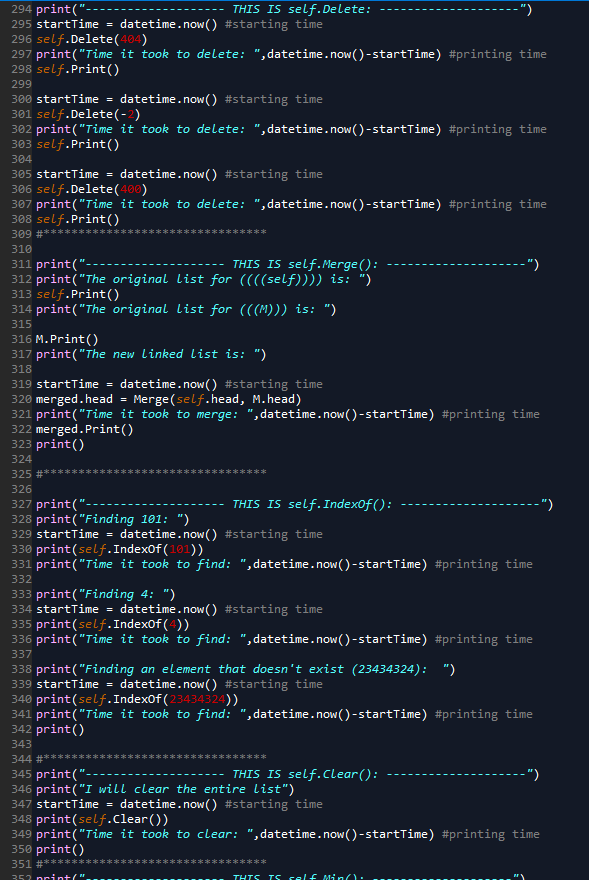
**Appendix**

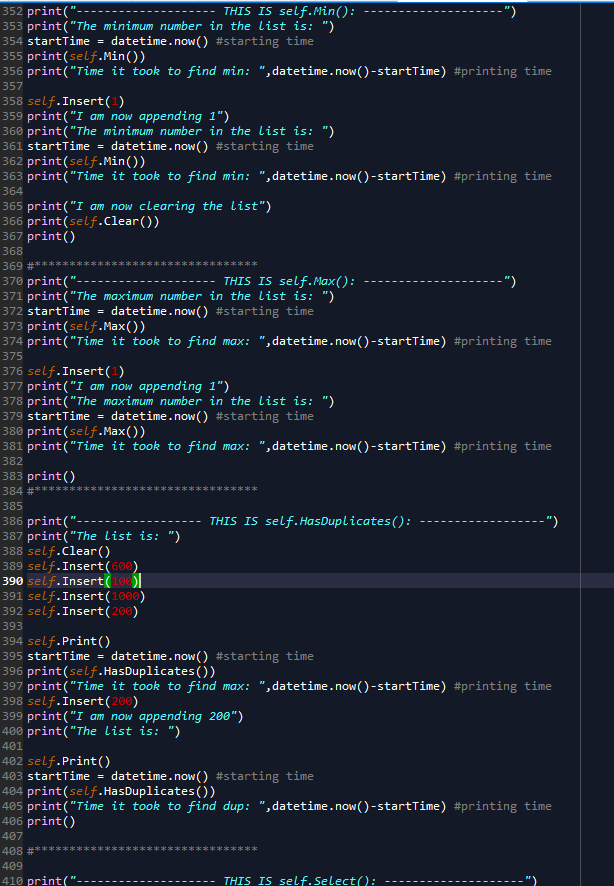
****

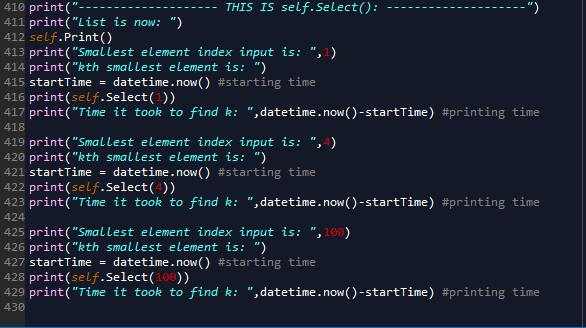
****

****

****

****

****

****

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.