CS 2302 Data Structures

Fall 2019

Lab Report #3

Due: October 4, 2019

Professor: Olac Fuentes

TA: Anindita Nath

**Introduction**

In this lab the task was to implement different methods That would edit singly linked lists or reference base list in ways listed from the lab assignment. this lab was meant to serve as both a refresher from the computer science true course as well as dive a little deeper into how reference base lists actually function. it also helped me a little more with the way classes interact with each other on a fundamental level and taught me a little bit of python in the process.

**Proposed Solution Design and Implementation**

For the majority of this lab I wanted to try a different method of actually approaching each method. The method that I wanted to try was to develop working methods that were not necessarily the most efficient but worked as instructed in the laugh out loud. once I made methods that worked I would rather then go back and review each method and apply varying and different ways of trying to optimize the methods as well as adding checks that tend to cover the extreme edge cases. Once the program is finished I plan to apply each method sequentially editing the same list to verify functionality each method.

**Experimental Results**

**Print():**

This part of the lab was Easy to implement. All I needed to do was run through the list and as I got through each element in the list I would just simply print the data of that node.

TEST 1 - [0,1,2,3,4,5,6,7,8,9]



TEST 2 - [0,2,4,6,8,10]



TEST 3 - [3,4,5,6,7,8,9]



**insert():**

This part of the lab was For some reason the hardest part of the entire lab for me. I just fundamentally misunderstood how inserting into a already made list worked so I did need to relearn reference base list which proved challenging but after I did it was fairly simple. All I did was implemented check to see where anode be long as it rated through the list.

TEST 1 - [1,3,2,4,6,5,6]



TEST 2 - [8,7,6,5,4,3,2,1]



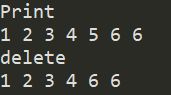
TEST 3 - [10,10,10,5,3,8,10]

****

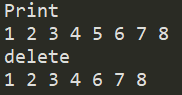
**delete():**

This one was very easy as well. all you needed to do was iterate through the list and once you reached the desired no to erase, you would just link the previous node with the next node of the node you wish to delete.

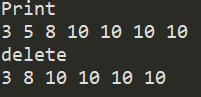
TEST 1 - [1,3,2,4,6,5,6] , delete(5)



TEST 2 - [8,7,6,5,4,3,2,1], delete(5)



TEST 3 - [10,10,10,5,3,8,10], delete(5)



**merge():**

This one was done simply by iterating through the second list that was given and as I would run through each node in that list I would insert that data into the main list which would merge the two lists because my insert function Auto sorts as it inserts.

TEST 1 - [1,3,2,4,6,5,6] , [8,7,6,5,4,3,2,1]



TEST 2 - [8,7,6,5,4,3,2,1], [10,10,10,5,3,8,10]



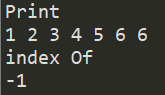
TEST 3 - [10,10,10,5,3,8,10], [1,3,2,4,6,5,6]



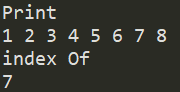
**indexOf():**

This one all I did was iterate through the list and for each know that I passed through I would increase a counter and once I reached the desired node I will just return the counter otherwise if I made it to the end I would just return -1.

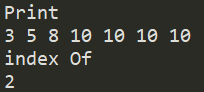
TEST 1 - [1,3,2,4,6,5,6], indexOf(8)



TEST 2 - [8,7,6,5,4,3,2,1], indexOf(8)



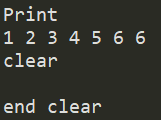
TEST 3 - [10,10,10,5,3,8,10]], indexOf(8)



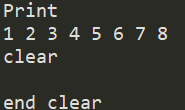
**clear():**

Another simple one. Just set both the head and the tail to null.

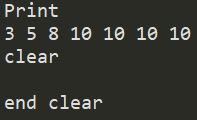
TEST 1 - [1,3,2,4,6,5,6]



TEST 2 - [8,7,6,5,4,3,2,1]



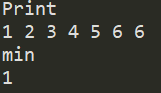
TEST 3 - [10,10,10,5,3,8,10]



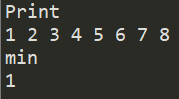
**min():**

In a sorted list this is easy. Just return the very first element.

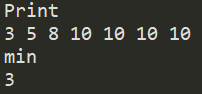
TEST 1 - [1,3,2,4,6,5,6]



TEST 2 - [8,7,6,5,4,3,2,1]



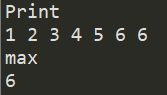
TEST 3 - [10,10,10,5,3,8,10]



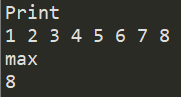
**max():**

In a sorted list this is easy. Just return the very last element.

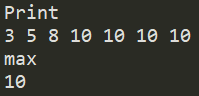
TEST 1 - [1,3,2,4,6,5,6]



TEST 2 - [8,7,6,5,4,3,2,1]



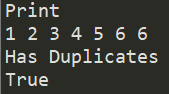
TEST 3 - [10,10,10,5,3,8,10]



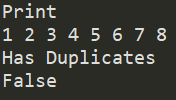
**hadDuplicates():**

For this one I just iterated through the list and for each node i iterated through I would add it to a set that I would reference throughout the entire iteration of the list.and if the node at the current moment of iteration was in the reference set I would return true because that means there was a duplicate in the list.

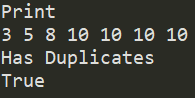
TEST 1 - [1,3,2,4,6,5,6]



TEST 2 - [8,7,6,5,4,3,2,1]



TEST 3 - [10,10,10,5,3,8,10]



**Overall Lab Results:**

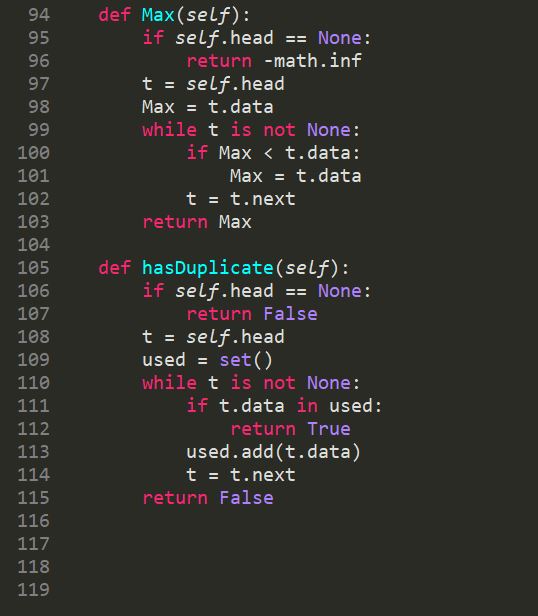
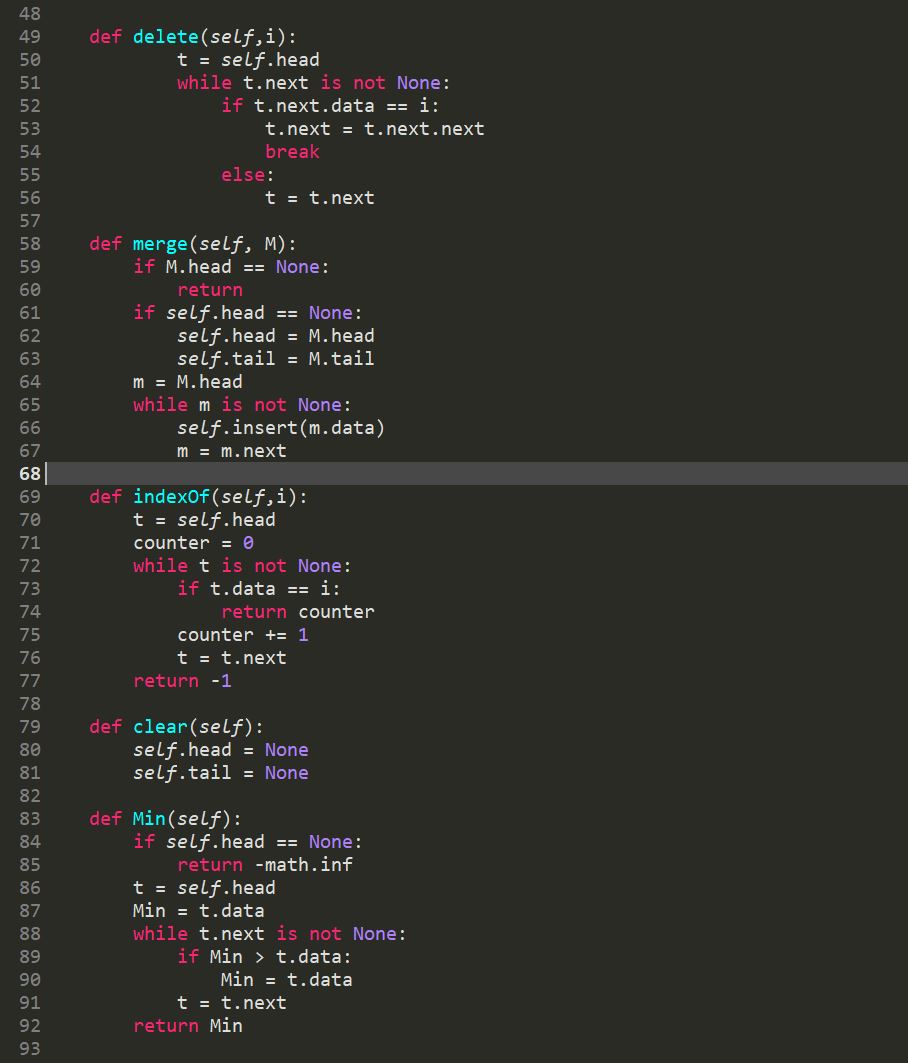
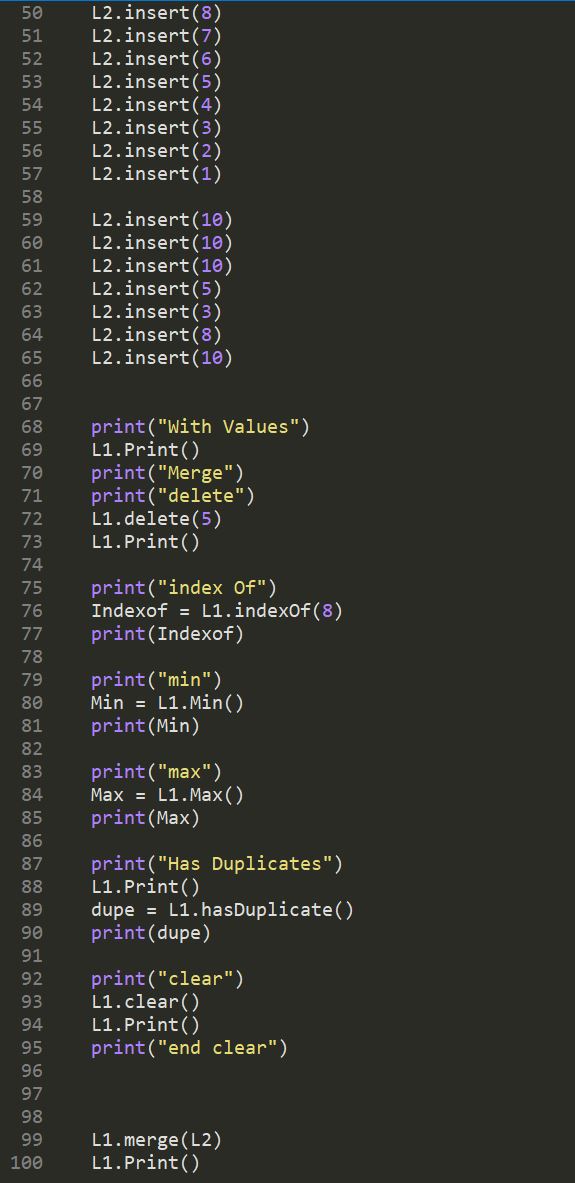
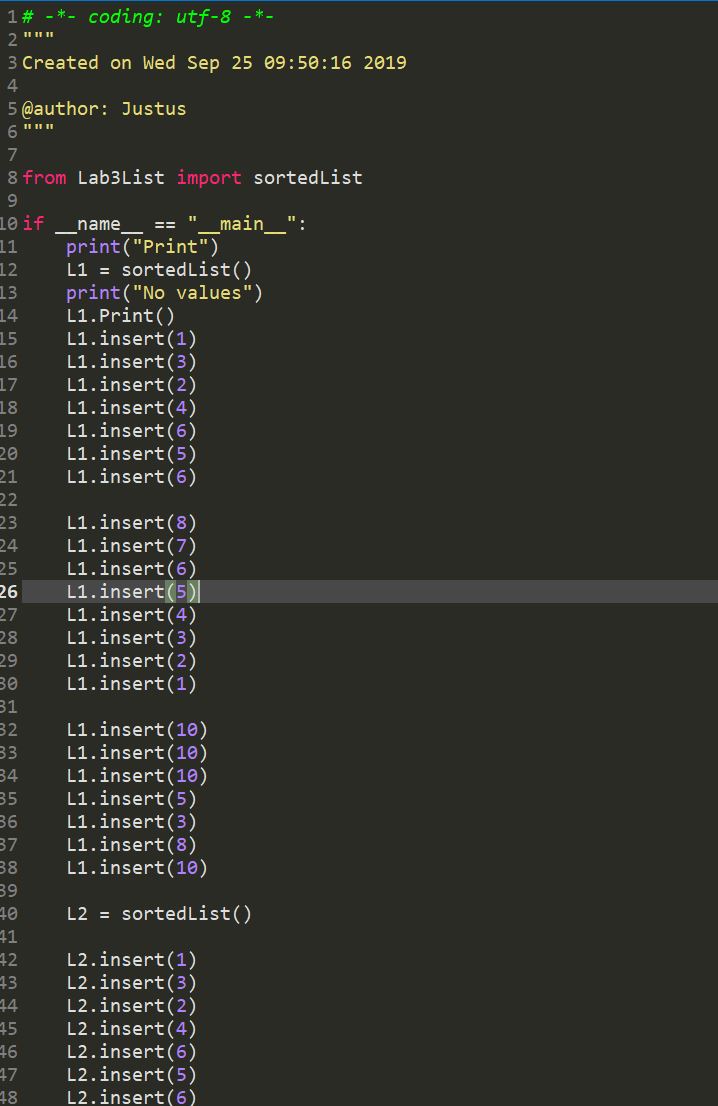
After going back and running through each of the methods trying to find different optimizations below is a chart that compares the run times with the methods I created versus how those methods would perform in a non sorted list environment in BigO Notation.

|  |  |  |
| --- | --- | --- |
| Function | SortedList | List |
| Print() | O(n) | O(n) |
| Insert(i) | O(n) | O(1) |
| Delete(i) | O(1) | O(n) |
| Merge(M) | O(n^2) | O(1) |
| IndexOf(i) | O(n) | O(n) |
| Clear(i) | O(1) | O(1) |
| Min() | O(1) | O(n) |
| Max() | O(1) | O(n) |
| HasDupes() | O(n) | O(n) |

**Conclusion**

I've always been fairly confident with reference base list so this lab was a nice break from fundamental concepts that I still don't fully grasp that require a lot of brain processing power. there's still a lot for me to work on to fully utilize my ability to implement reference base list.

**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*

Justus Frausto