CS2302 Data Structures

Fall 2019

Lab Report #3

Due October 4th, 2019

Professor: Olac Fuentes

TA: Anindita Nath

**Introduction**

For this assignment we had to implement various methods that would do things like insert an integer into a linked list, delete an integer from a linked list, find the min or max element, see if there was duplicate integers in a linked list, clear a linked list, merge linked lists, and give the index of an element .

**Proposed Solution Design and Implementation**

I created each method separately and called them from the main. The following was the implementation of each method:

Print: I would simply traverse the linked list using a while loop and print the elements of the list as I go.

Insert: If the list I empty I make a new node assign the integer to the node and set it as the head and tail, otherwise I insert the element at the end and set it to be the tail.

Delete: I the list is empty I do nothing, otherwise I create two trackers, one that is set to my the head and the other one position ahead which is head.next. I traverse the list and compare to see if the data in the head.next tracker is the same as the integer I want to remove. If it is, I use my current position and set the next position to be .next.next, which removes the element from the list.

Merge: I set the tail.next from list self to be the head of the list M.

IndexOf: if the list is empty I return -1. Otherwise, I traverse the list until the element I am in has the same integer as the one passed in the method and I return the element.

Clear: I set self.head and self.tail to be None

Min: if the list is empty I return math.inf. Otherwise, I traverse the list and make a comparison for each element I traverse. I the element is our new min is set the min variable to be the current element I am in.

Max: if the list is empty I return math.inf. Otherwise, I traverse the list and make a comparison for each element I traverse. I the element is our new max is set the max variable to be the current element I am in.

HasDuplicates: I traverse the list, since it is sorted I compare our current position to our next position. If they match I return true if they don’t I return false.

Select: This had the same implementation as lab 2 I simply traverse the sorted list and use a counter, once the counter is the same as our kth – 1 element I return the element of our current position.

**Experimental Results**

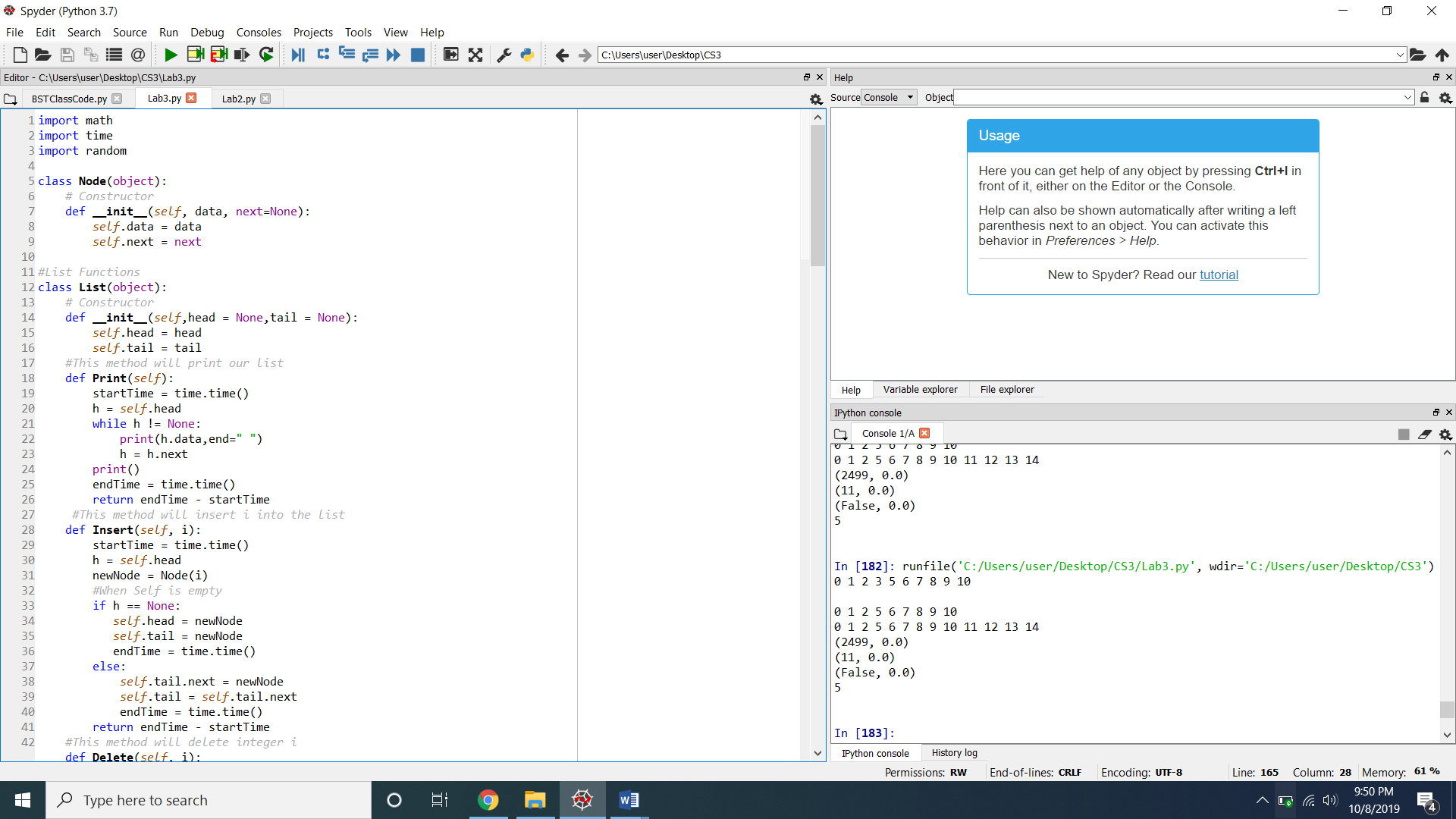
|  |  |  |
| --- | --- | --- |
| Function | SortedList | List |
| Print | O(n) | O(n) |
| Insert | O(n) | O(n) |
| Delete | O(n) | O(n) |
| Merge | O(1) | O(1) |
| IndexOf | O(n) | O(n) |
| Clear | O(1) | O(1) |
| Min | O(n^2) | O(n^2) |
| Max | O(n^2) | O(n^2) |
| Duplicates | O(n^2) | O(n^2) |
| Selection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

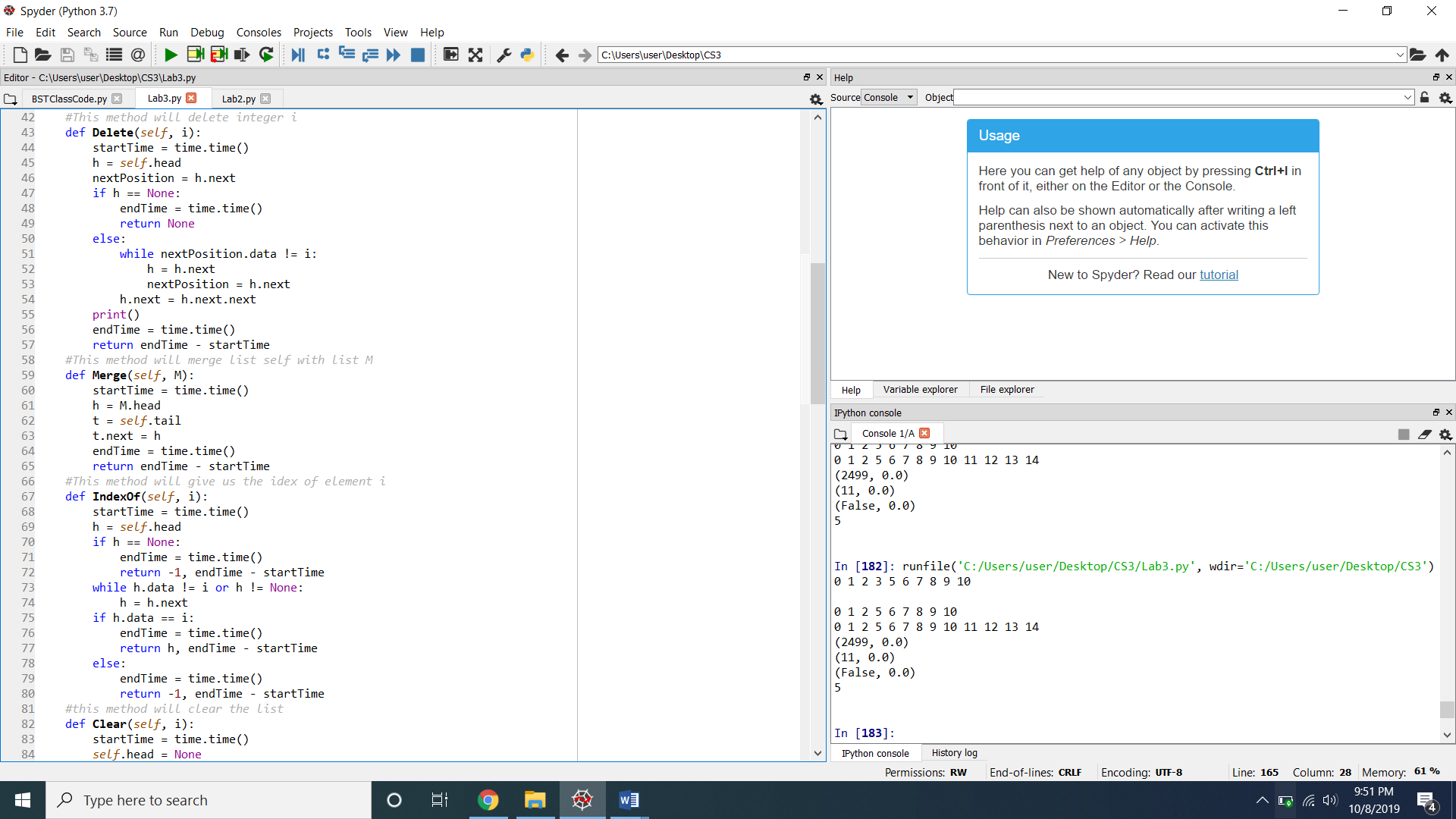
**Conclusion**

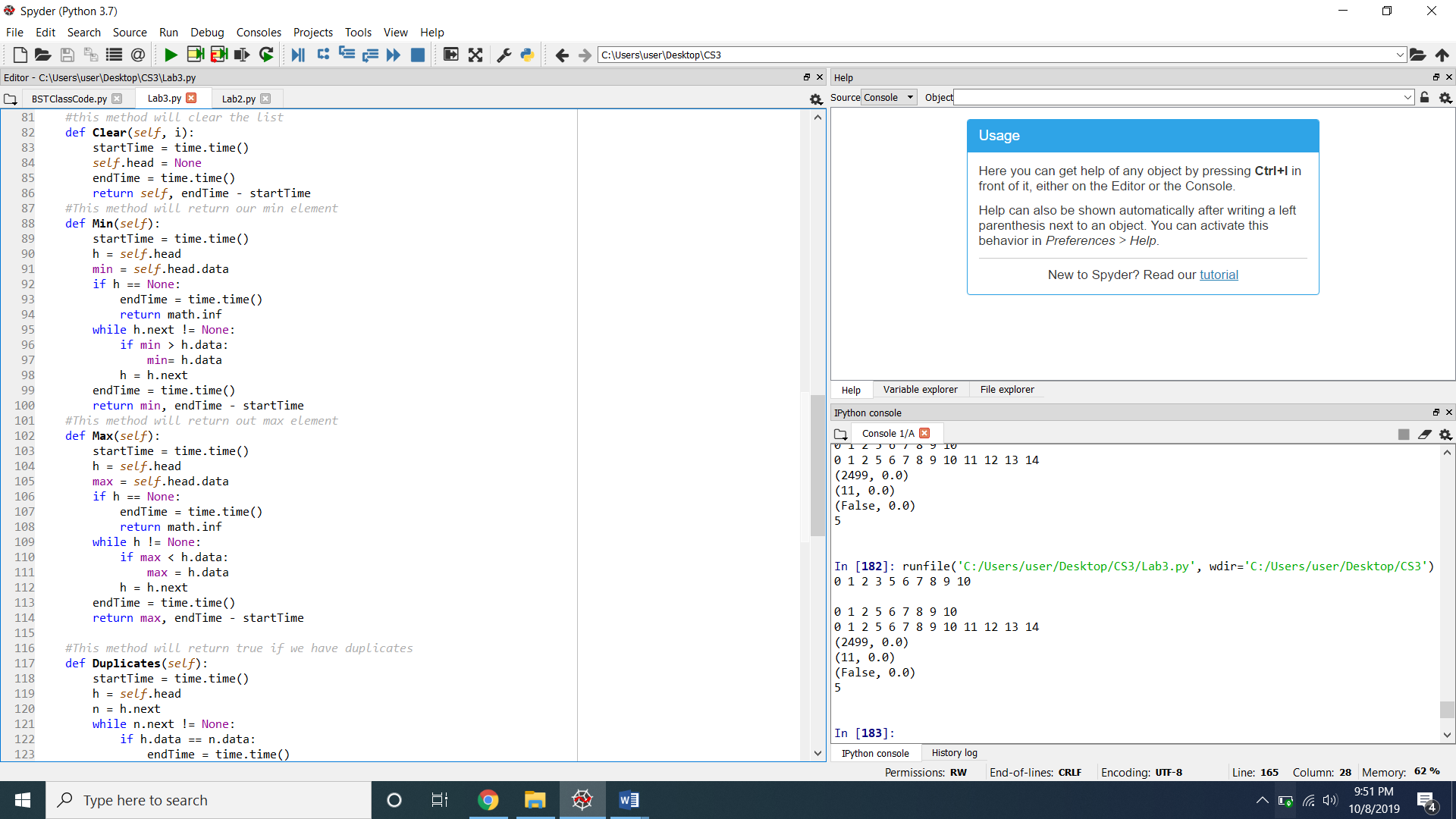
This lab helped me understand how to implement various methods for a linked list. This lab was challenging for me and took me a long time to implement. I had to use many concepts learned in CS2, which were very helpful in this Lab. I also learned how to decide which tests I should run for each method to get the best and worst case scenarios.

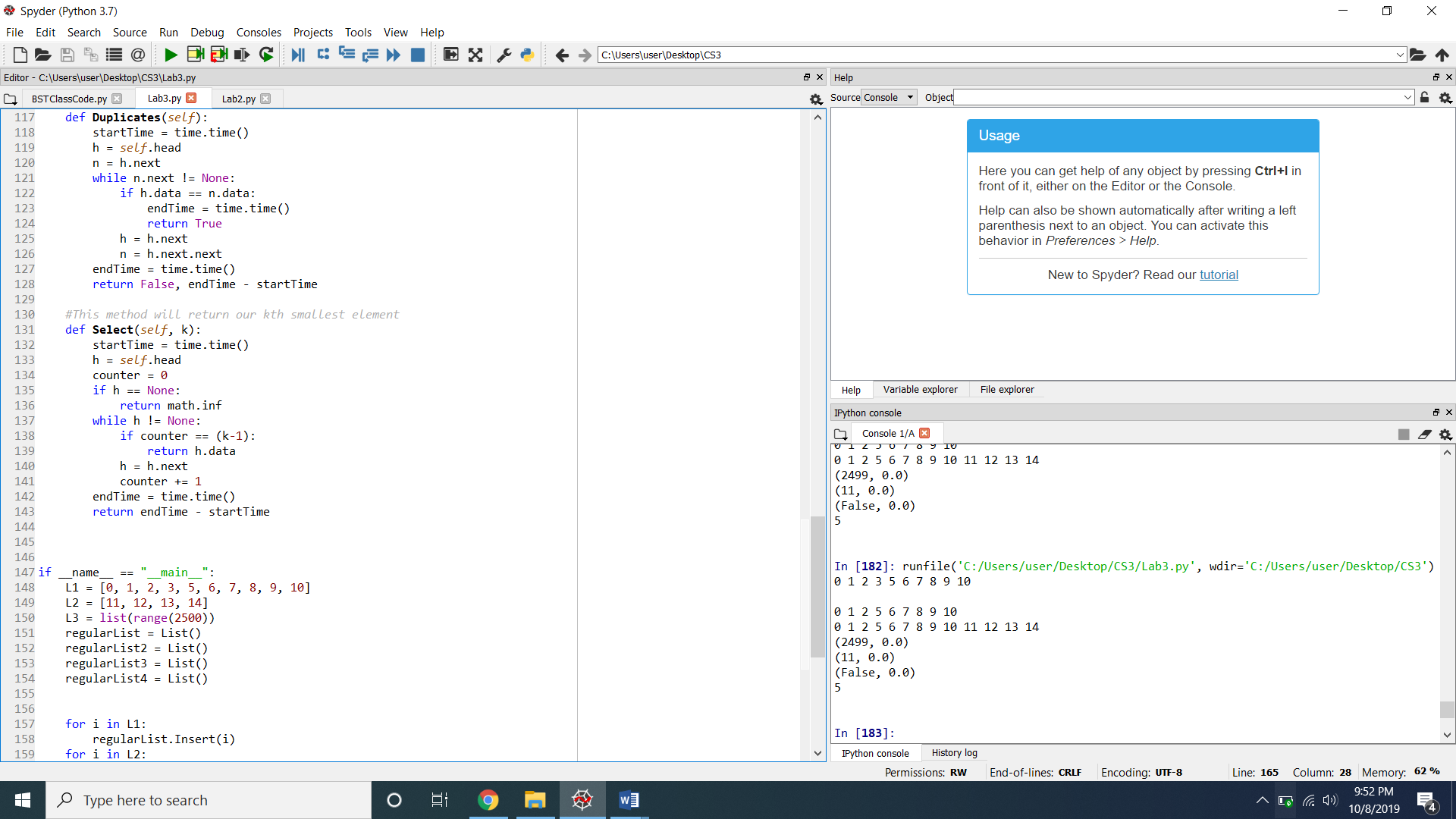
**Appendix**

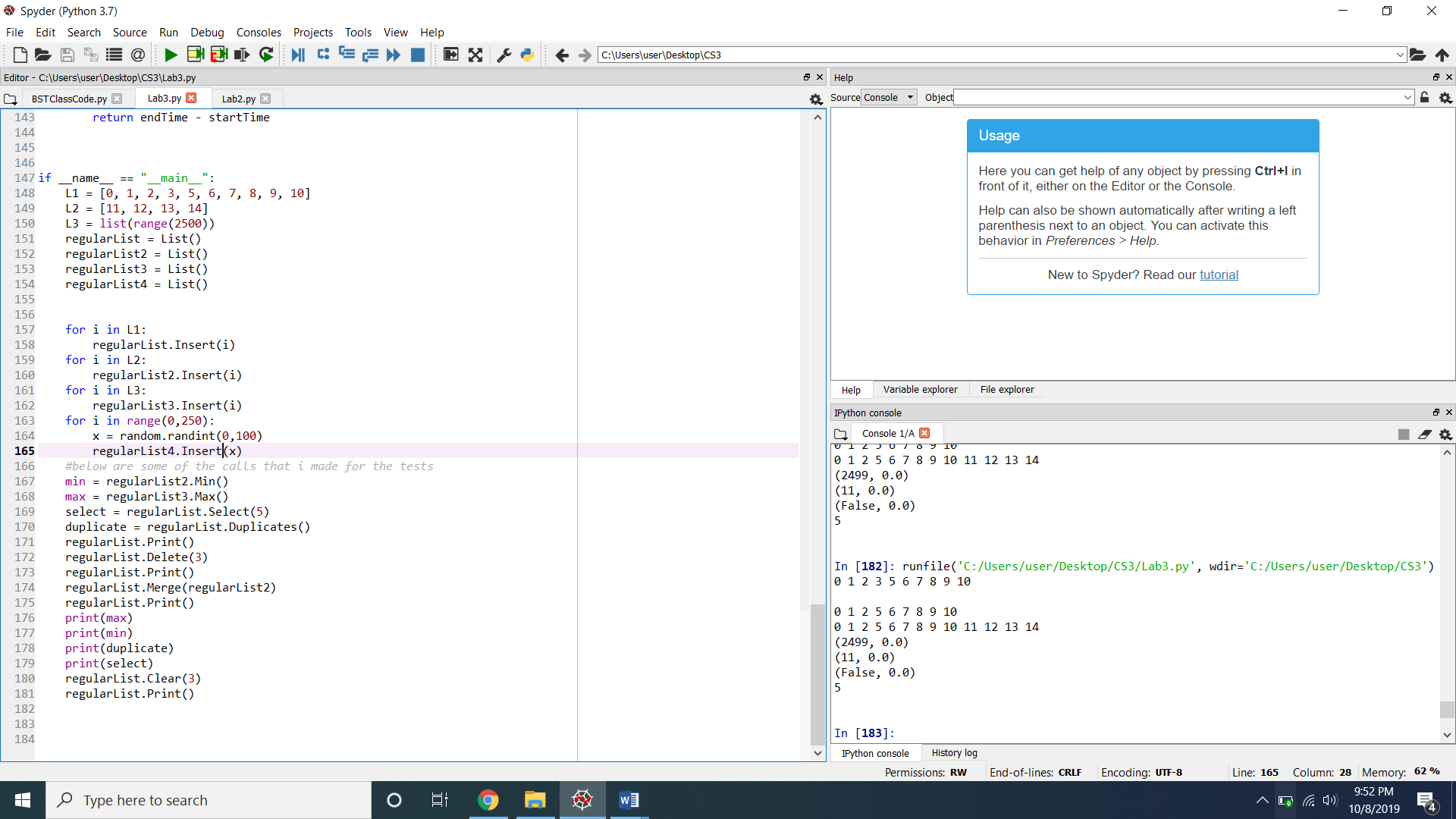
The following is my source code











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