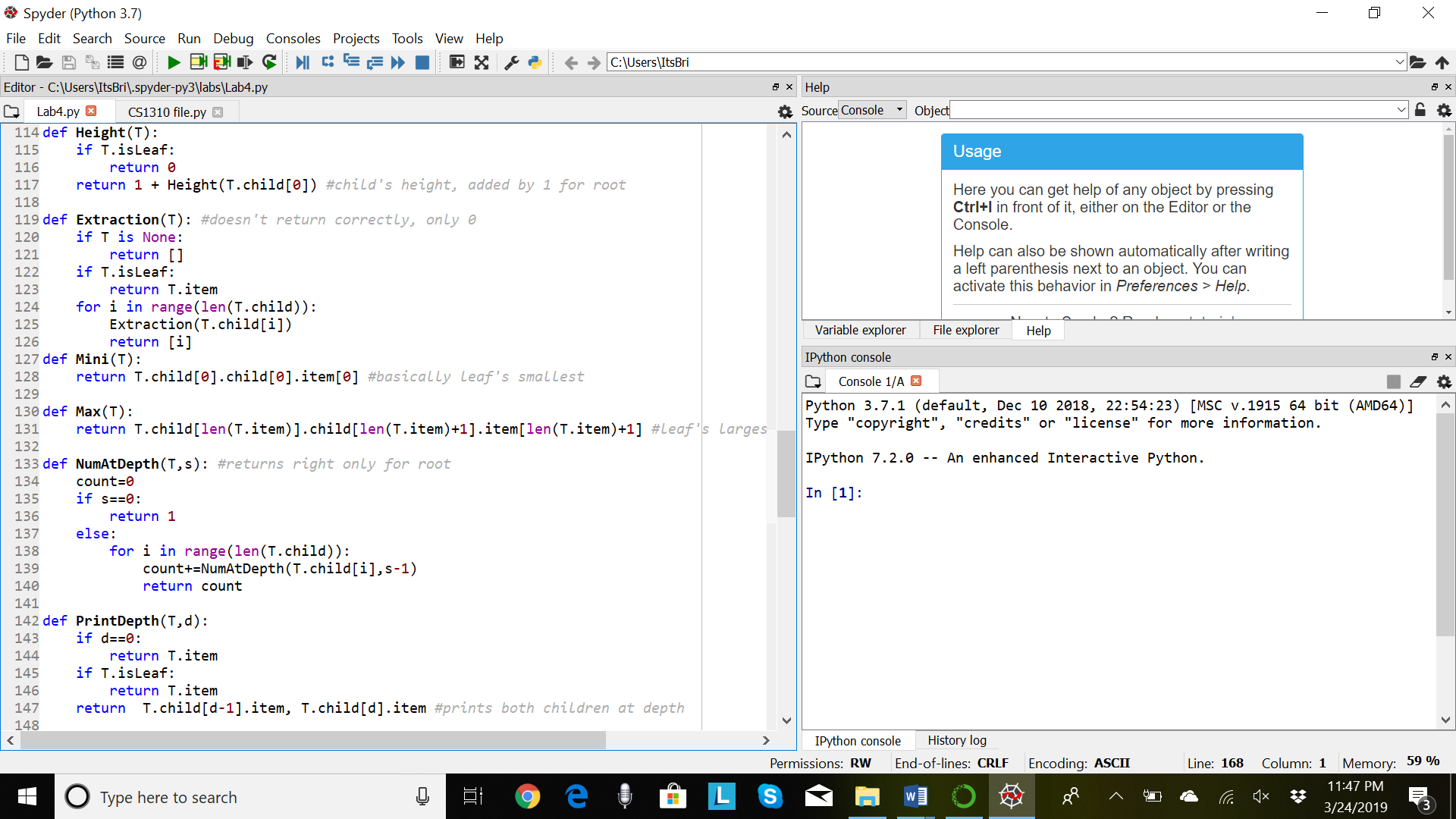
Report Lab 4

This lab was easier to do than last lab, at least for me. Overall, creating the methods for this balanced tree was only slightly difficult.

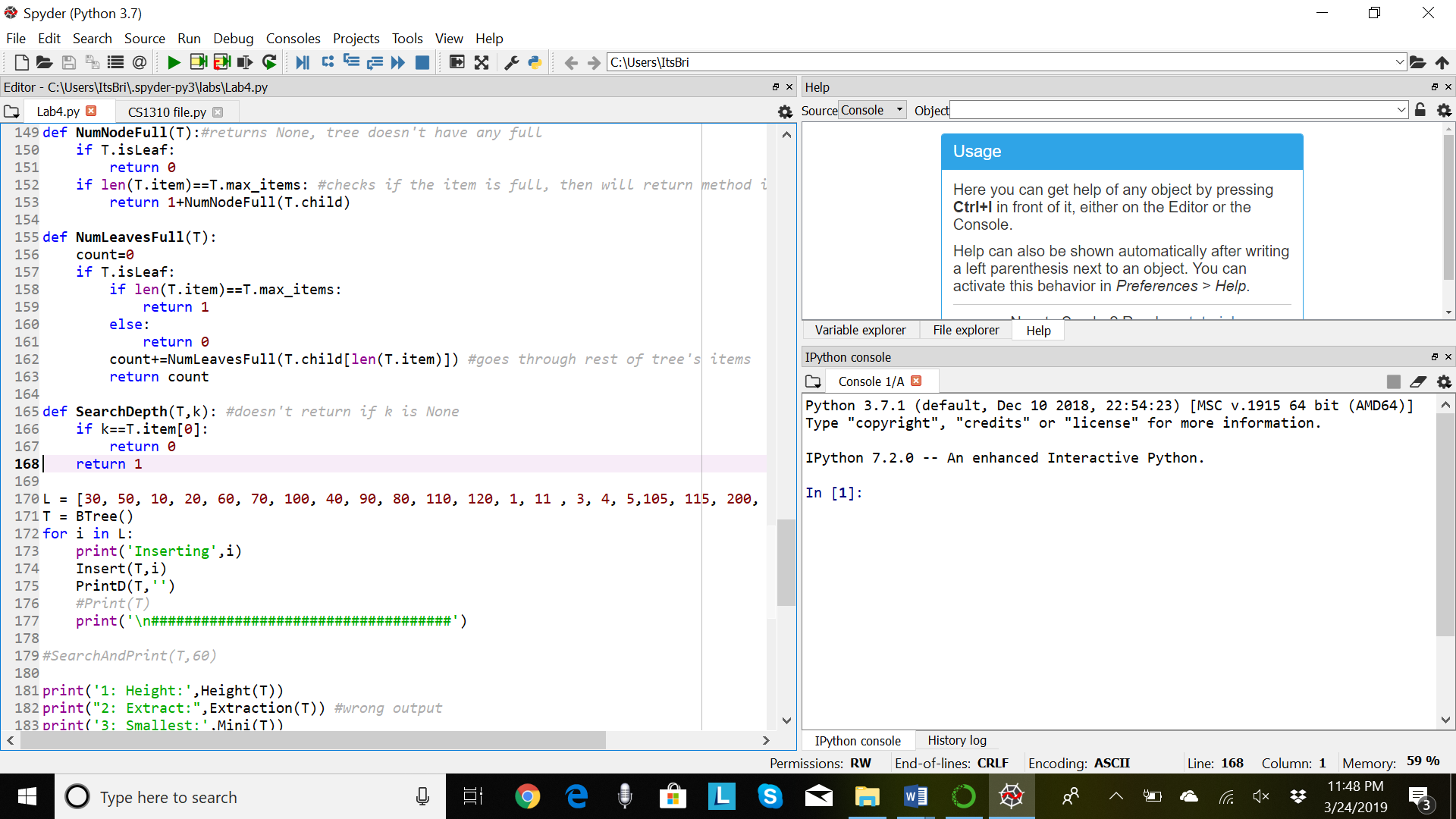
For my first method, it was already given by the professor, but I took time to understand what this part of code meant. For my second method, Extraction, is for extracting the nodes in the tree into a list and print that list out. I was able to return 0 but couldn’t return any other nodes given. I tried changing my if statements, such as if leaf or if it was empty, and they followed correctly, but my For loop wasn’t going through each item to print ‘i’ of what each is.

My third method was very simple. I took the first item, of the first child, of the first child, to return the smallest item. It was very specific to this code but can work for a handful.

Fourth method was as easy as the third, but instead used the length of the item, of the child, of the child, to return the biggest item in the BT.

My fifth method was one of the last methods I worked on. I tried another For loop to go through each item in the depth given to return the Count of how many it found. Unfortunately, only one if statement worked, which was for the root. The root is returned correctly, 1, but not for a higher level.

The sixth method needed some work. It follows two if statements, then returns the child at depth. I had to code this weirdly for it to return all the items at the depth, but I see now that my depth ‘d’ shouldn’t be held in place for the number item for the child.



Methods seventh and eighth took the longest to work on. They both follow a similar structure of checking if Leaf and returning the given amount of either nodes full or nodes that are leaves. They each return, following the if statements, their corresponding amounts. Not much to expand on.

Lastly, the ninth method to Search for the given item, and return its depth. The first statement is for if the item is the root, which returns 0. It’ll return 1 if the item is in the next depth. The flaw I forgot to fix was it doesn’t return -1 if the k, item, is not there or None.

Overall, my design for this code is direct and specific. I tried experimenting with different ideas of code that might’ve worked for all types of BTs but couldn’t have a solid method made. Each of my methods have different structures, but most are specific to the BT we were provided with.

Honesty Statement:

Academic dishonesty includes but is not limited to cheating, plagiarism and collusion. Cheating may involve

copying from or providing information to another student, possessing unauthorized materials during a test, or

falsifying data (for example program outputs) in laboratory reports. Plagiarism occurs when someone

represents the work or ideas of another person as his/her own. Collusion involves collaborating with another

person to commit an academically dishonest act. Professors are required to - and will - report academic

dishonesty and any other violation of the Standards of Conduct to the Dean of Students.

I hereby state hereby this code is mine and mine alone.

Appendix:

def Height(T):

if T. isLeaf:

return 0

return 1 + Height(T. child[0]) #child's height, added by 1 for root

def Extraction(T): #doesn't return correctly, only 0

if T is None:

return []

if T. isLeaf:

return T. item

for i in range(len(T. child)):

Extraction(T. child[i])

return [i]

def Mini(T):

return T. child[0].child[0].item[0] #basically leaf's smallest

def Max(T):

return T. child[len(T.item)].child[len(T.item)+1].item[len(T.item)+1] #leaf's largest

def NumAtDepth(T, s): #returns right only for root

count=0

if s==0:

return 1

else:

for i in range(len(T. child)):

count+=NumAtDepth(T. child[i],s-1)

return count

def PrintDepth(T, d):

if d==0:

return T. item

if T. isLeaf:

return T. item

return T.child[d-1].item, T.child[d].item #prints both children at depth

def NumNodeFull(T):#returns None, tree doesn't have any full

if T. isLeaf:

return 0

if len(T. item)==T.max\_items: #checks if the item is full, then will return method if is to check the rest

return 1+NumNodeFull(T. child)

def NumLeavesFull(T):

count=0

if T. isLeaf:

if len(T. item)==T.max\_items:

return 1

else:

return 0

count+=NumLeavesFull(T. child[len(T.item)]) #goes through rest of tree's items

return count

def SearchDepth(T, k): #doesn't return if k is None

if k==T.item[0]:

return 0

return 1