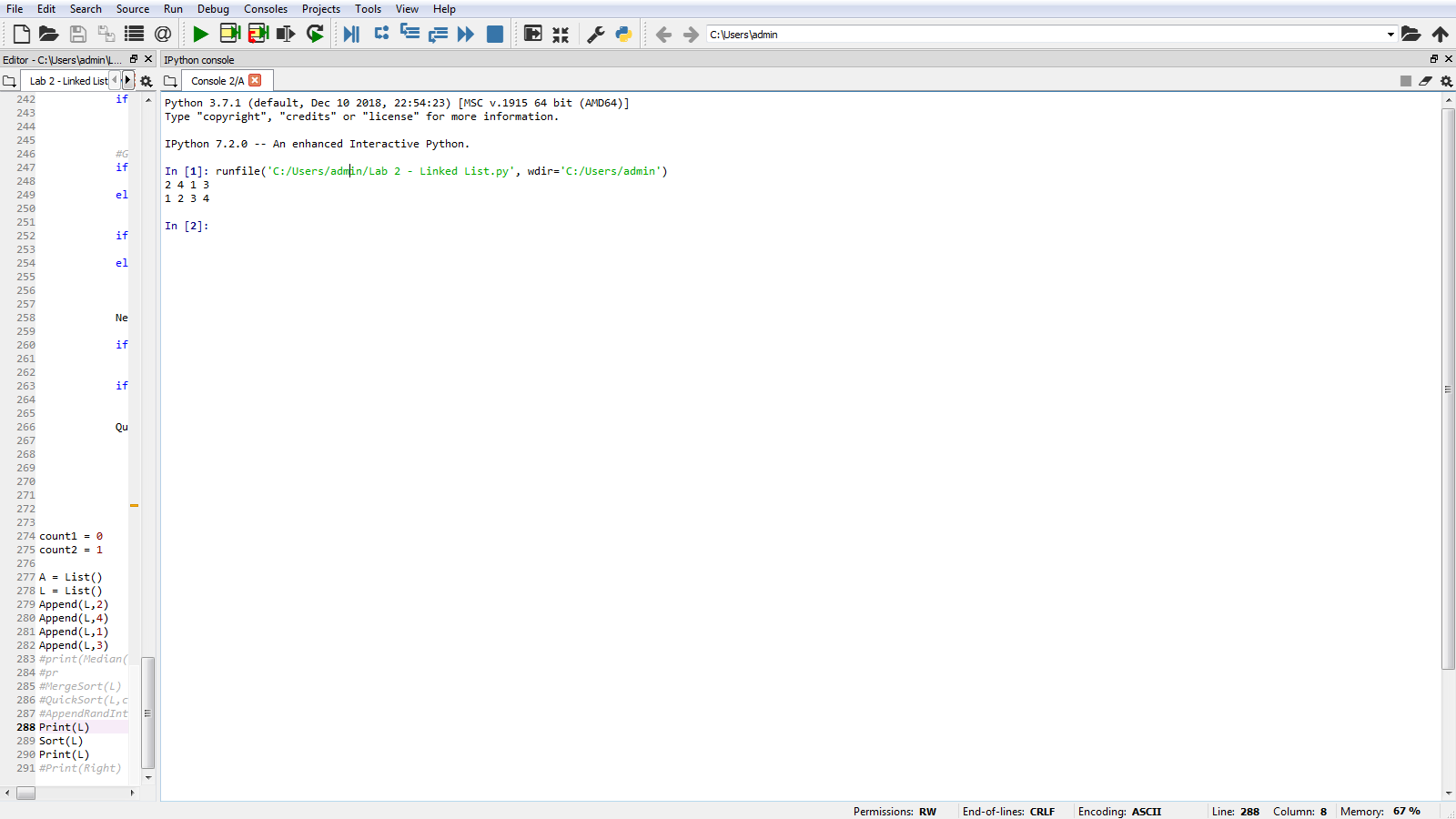
Solomon Davis Lab 2 Report

CS 2303 – MW 1:30 Spring 2019

For lab 2 we worked on the sorting list for bubble sort, merge sort, and quicksort. I was unable to get the quicksort or merge sort to work. I have trouble with the list for both. For the bubble sort I compared two values at a time if the second value was bigger than the first, I switched the order by creating a temporary value and using the temp to switch the values of each variable. I was also unable to get the random int and median working. I was able to calculate the median but was unable to get the index of the list I wanted. For the merge sort methods, I tried to split in two continuously but had issues after the first list. For the quicksort like merge sort was split the list by the pivot value into two lists, one bigger than the pivot and one smaller. I tried to recursively call the method for each list but had trouble with the list after getting the first list.

This the output for me appending items to a list and then sorting them using bubble sort



#Course: CS2302 - Spring 2019

#Author: Solomon Davis

#Lab Number: 2

#Instructor: Olac Fuentes

#Last Modified: February 22, 2019

#Due Date: February 22, 2019

#Description: This code was supposed generate a random sized list with random

# integers by using the merge sort, bubble sort, and merge sort and

# return the median.

import random

#Node Functions

class Node(object):

# Constructor

def \_\_init\_\_(self, item, next=None):

self.item = item

self.next = next

def PrintNodes(N):

if N != None:

print(N.item, end=' ')

PrintNodes(N.next)

def PrintNodesReverse(N):

if N != None:

PrintNodesReverse(N.next)

print(N.item, end=' ')

#List Functions

class List(object):

# Constructor

def \_\_init\_\_(self):

self.head = None

self.tail = None

def IsEmpty(L):

return L.head == None

def Append(L,x):

# Inserts x at end of list L

if IsEmpty(L):

L.head = Node(x)

L.tail = L.head

else:

L.tail.next = Node(x)

L.tail = L.tail.next

def Prepend(L,x):

# Inserts x at beginning of list L

if IsEmpty(L):

L.head = Node(x)

L.tail = L.head

else:

L.head = Node(x, L.head)

def Concatenate(L,L2):

if IsEmpty(L):

L.head = L2.head

L.tail = L2.head

else:

if not IsEmpty(L2):

L.tail.next = L2.head

L.tail = L2.tail

def AppendRandInts(L):

i = random.randInt(0,9)

for i in range(10):

Append(L,random.randInt(0,9))

def Print(L):

# Prints list L's items in order using a loop

temp = L.head

while temp is not None:

print(temp.item, end=' ')

temp = temp.next

print() # New line

def PrintRec(L):

# Prints list L's items in order using recursion

PrintNodes(L.head)

print()

def Remove(L,x):

# Removes x from list L

# It does nothing if x is not in L

if L.head==None:

return

if L.head.item == x:

if L.head == L.tail: # x is the only element in list

L.head = None

L.tail = None

else:

L.head = L.head.next

else:

# Find x

temp = L.head

while temp.next != None and temp.next.item !=x:

temp = temp.next

if temp.next != None: # x was found

if temp.next == L.tail: # x is the last node

L.tail = temp

L.tail.next = None

else:

temp.next = temp.next.next

def GetLength(L):

temp = L.head

count = 0

while temp is not None:

count +=1

temp = temp.next

return count

def PrintReverse(L):

# Prints list L's items in reverse order

PrintNodesReverse(L.head)

print()

def Copy(A):

temp = L.head

while temp is not None:

Append(A,temp.item)

temp = temp.next

#Print(A)

#C = Copy(C)

#Print(Copy)

return A

def Search(L,x):

if IsEmpty(L):

L.head = Node(x)

L.tail = L.head

else:

temp = L.head

while temp is not None:

if temp.item == x:

temp = temp.next

def InsertAfter(L,x,item):

if IsEmpty(L):

L.head = Node(x)

L.tail = L.head

else:

temp = L.head

while temp is not None:

if temp.item == x:

temp.next = Node(item,temp.next)

temp = temp.next

def Median(L):

# C = Copy(L)

BubbleSort(L)

return L[(GetLength(L)//2)]

def BubbleSort(L):

#Running Time = O(n^2)

c = True

while c:

temp = L.head

c = False

while temp.next is not None:

if temp.item > temp.next.item:

tempValue = temp.item

temp.item = temp.next.item

temp.next.item = tempValue

c = True

temp = temp.next

def MergeSort(L):

#Merge Sort

#Running Time = O(nlogn)

Left = List()

Right = List()

temp = L.head

while temp is not None:

Append(Left,temp.item)

temp = temp.next

if temp is not None:

Append(Right,temp.item)

temp = temp.next

if GetLength(Left) is not 1:

MergeSort(Left)

if GetLength(Right) is not 1:

MergeSort(Right)

print(temp.item)

def MergeSort2(NewList):

t1 = Left.head

t2 = Right.head

while t1 is not None and t2 is not None:

if t1.item < t2.item:

Append(NewList,t1.item)

t1 = t1.next

if t1.item > t2.item:

Append(NewList,t2.item)

t2 = t2.next

else:

Append(NewList,t1.item)

Append(NewList,t2.item)

t1 = t1.next

t2 = t2.next

if t1 is None:

while t2 is not None:

Append(NewList,t2.item)

t2 = t2.next

if t2 is None:

while t1 is not None:

Append(NewList,t1.item)

t1 = t1.next

def QuickSort(L,count1,count2):

#running time

#Ologn

temp = L.head

pivot = List()

pivot.next = temp.item

temp = temp.next

smallerList = List()

largerList = List()

while temp.next is not None:

if temp.next.item < pivot.next:

Append(smallerList,temp.next.item)

temp = temp.next

if temp.next.item > pivot.next:

Append(largerList,temp.next.item)

temp = temp.next

#GetLength(smallerList)

if GetLength(smallerList) == 1:

print("Smaller List ",count1,"is finished")

else:

QuickSort(smallerList[count1:count2],count1+1,count2+1)

if GetLength(largerList) == 1:

print("Larger List ",count1,"is finished")

else:

QuickSort(largerList[count1:count2],count1+1,count2+1)

#start Comparison of list values

NewList = List()

if smallerList[0:1] < pivot.next:

InsertAfter(NewList,pivot.next,smallerList[count1:count2])

if smallerList[count1:count2] > pivot.next:

Prepend(NewList,pivot.next,smallerList[count1:count2])

QuickSort(NewList,count1-1,count2-1)

#while count > 0

# if temp2.next < pivot[count]:

# Prepend(NewList)

#if temp2.next > pivot[count]:

# Append(NewList)

#count -=1

count1 = 0

count2 = 1

NewList = List()

Left = List()

Right = List()

A = List()

L = List()

Append(L,2)

Append(L,4)

Append(L,1)

Append(L,3)

#print(Median(L))

#pr

#MergeSort(L)

#QuickSort(L,count1,count2)

#AppendRandInts(L)

#Print(NewList)

BubbleSort(L)

Print(L)

#MergeSort(L)

#Print(Left)

#Print(Right)

#Print(NewList)

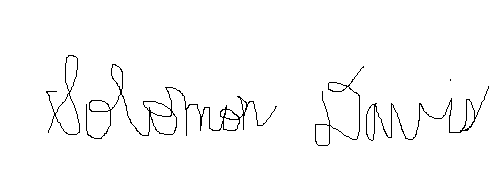
#Sort(L)

#Print(L)

#Print(Right)

Academic Service Certificate:

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



Solomon Davis