Sorting it all out

LAB # 7

By

Corey Henry

And

Aaron Shepard

***“On my honor, as a Mississippi State University student, I have neither***

***given nor received unauthorized assistance on this academic work.”***

Signature:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

CSE-1384-06-201430 Intermediate Computer Programming

Class Section # 6

Jesse Farek

2/17/2015

**Analysis and Conclusions**

The code was very easy the point was to create multiple types of functions using recursion. Most of these were able to be done very easily with loops or built in functions but we could not use them. The min and max function were very easy and only required you to change the greater or less than sign depending on which one you wanted. The addition function was also easy we just took the first element in the list and added each other index in the list till the end. Quicksort was provided in our class lecture as well as the max function. Printing was easy, we set to list equal to a string formatted version and removed the brackets and the returned the string version of the list.

Source Code:

Corey Henry & Aaron Shepard #Date Assigned: 17 mar 15

# #

#Course CSE 1384 Sec 06 #Date Due: 24 mar 15

#File name: Lab

#

#Program description- recursice functions for max,min, print, quicksort.

# function that will take the list and find the maxiumum value out of the list

# and returns the highest

def recursive\_max(my\_list):

if len(my\_list) == 1:

return my\_list[0]

else:

biggest = recursive\_max(my\_list[1:])

if biggest > my\_list[0]:

return biggest

else:

return my\_list[0]

# takes the first index of the list and adds the rest of the list onto the first index

# uses recursice to keep adding the numbers

def recursive\_sum(my\_list):

if len(my\_list) == 1:

return my\_list[0]

else:

return my\_list[0] + recursive\_sum(my\_list[1:])

# uses the same function as the min you just switch the signs, finds the lowest

# value in the list and returns it.

def recursive\_min(my\_list):

if len(my\_list) ==1:

return my\_list[0]

else:

smallest = recursive\_min(my\_list[1:])

if smallest < my\_list[0]:

return smallest

else:

return my\_list[0]

#prints the list starting with the first index and goes through, when it reaches

# it will print the last index in the list

def recursive\_print(my\_list):

if len(my\_list) == 1:

print(my\_list[0])

else:

print(my\_list[0], end = ' ')

recursive\_print(my\_list[1:])

#taken from class slides. takes the middle number and splits the list in two.

def quicksort(my\_list, first, last):

if first < last - 1:

pivot = median3(my\_list, first, last)

i = first

j = last -1

while True:

i+=1

while my\_list[i] < pivot:

i +=1

j -= 1

while my\_list[j] > pivot:

j -=1

if i < j:

my\_list[i], my\_list[j], = my\_list[j], my\_list[i]

else:

break

my\_list[i], my\_list[last - 1] = my\_list[last -1], my\_list[i]

quicksort(my\_list, first, i -1)

quicksort(my\_list, i+1, last)

elif first < last:

if my\_list[first] > my\_list[last]:

my\_list[first], my\_list[last] = my\_list[last], my\_list[first]

def median3(my\_list, first, last):

center = (first + last) //2

if my\_list[center] < my\_list[first]:

my\_list[first], my\_list[center] = my\_list[center], my\_list[first]

if my\_list[last] < my\_list[first]:

my\_list[first], my\_list[last] = my\_list[last], my\_list[first]

if my\_list[last] < my\_list[center]:

my\_list[center], my\_list[last] = my\_list[last], my\_list[center]

my\_list[center], my\_list[last -1] = my\_list[last-1], my\_list[center]

return my\_list[last-1]

