“Homework 14” Justin Minsk

1.

def lin\_search1(list, value):  
 i = len(list) - 1 # get i to the last index  
 while i != 0 and list[i] != value: # go through the list  
 i = i - 1 # look at next index  
 if i == 0: # if it goes through the list  
 return -1  
 else:  
 return i  
  
  
def lin\_search2(list, value):  
 for i in range(len(list) - 1, -1, -1): # go from last number to first  
 if list[i] == value: # if the list has the value  
 return i  
 return -1  
  
  
def lin\_search3(list, value):  
 list.insert(0, value) # insert the sent value  
 i = len(list) - 1 # make i at the end of the list  
 while list[i] != value: # go through the list  
 i = i - 1  
 list.pop(0)  
 if i == 0:  
 return -1  
 return i - 1  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 print(lin\_search1([0, 1 , 2, 3, 4], 4))  
 print(lin\_search2([0, 1 , 2, 3, 4], 4))  
 print(lin\_search3([0, 1 , 2, 3, 4], 4))

3.

# Nlog\_2 N + S \* Nlog\_2 N < S \* N  
# 2 \* log\_2 N < S

10.

def merge\_sort(x):  
 result = []  
 if len(x) < 2:  
 return x  
 mid = int(len(x) / 2)  
 y = merge\_sort(x[:mid])  
 z = merge\_sort(x[mid:])  
 i = 0  
 j = 0  
 while i < len(y) and j < len(z):  
 if y[i] > z[j]:  
 result.append(z[j])  
 j += 1  
 else:  
 result.append(y[i])  
 i += 1  
 result += y[i:]  
 result += z[j:]  
 return result  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 alist = [54, 26, 93, 17, 77, 31, 44, 55, 20]  
 print(merge\_sort(alist))

Extra Problem.

def list\_by\_mag(num\_list):  
 for nums in range(len(num\_list) - 1, 0, -1):  
 for i in range(nums):  
 if abs(num\_list[i]) > abs(num\_list[i + 1]):  
 temp = num\_list[i]  
 num\_list[i] = num\_list[i+1]  
 num\_list[i+1] = temp  
 if abs(num\_list[i]) == abs(num\_list[i + 1]):  
 if num\_list[i] > num\_list[i + 1]:  
 temp = num\_list[i]  
 num\_list[i] = num\_list[i + 1]  
 num\_list[i + 1] = temp  
 else:  
 temp = num\_list[i + 1]  
 num\_list[i + 1] = num\_list[i]  
 num\_list[i] = temp  
 if abs(num\_list[i]) == abs(num\_list[i - 1]):  
 if num\_list[i] > num\_list[i - 1]:  
 temp = num\_list[i]  
 num\_list[i] = num\_list[i - 1]  
 num\_list[i - 1] = temp  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 num\_list = [-2, -1, 2, 3, -5, 4, 2, -2, -4]  
 list\_by\_mag(num\_list)  
 print(num\_list)

import list\_by\_magnitude as ibm  
import unittest  
  
  
class Test\_list\_mag(unittest.TestCase):  
  
 def test\_list\_mag\_empty(self): # 1  
 *"""Test an empty list."""* argument = []  
 expected = []  
 ibm.list\_by\_mag(argument)  
 self.assertEqual(expected, argument, "The list is empty.")  
  
 def test\_running\_list\_mag\_item(self): # 2  
 *"""Test a one-item list."""* argument = [5]  
 expected = [5]  
 ibm.list\_by\_mag(argument)  
 self.assertEqual(expected, argument, "The list contains one item.")  
  
 def test\_running\_list\_mag\_items(self): # 3  
 *"""Test a two-item list."""* argument = [2, 5]  
 expected = [2, 5]  
 ibm.list\_by\_mag(argument)  
 self.assertEqual(expected, argument, "The list contains two items.")  
  
 def test\_running\_list\_mag\_negative(self): # 4  
 *"""Test a list of negative values."""* argument = [-1, -5, -3, -4]  
 expected = [-1, -3, -4, -5]  
 ibm.list\_by\_mag(argument)  
 self.assertEqual(expected, argument, "The list contains only negative values.")  
  
 def test\_running\_list\_mag\_zeros(self): # 5  
 *"""Test a list of zeros."""* argument = [0, 0, 0, 0]  
 expected = [0, 0, 0, 0]  
 ibm.list\_by\_mag(argument)  
 self.assertEqual(expected, argument, "The list contains only zeros.")  
  
 def test\_running\_list\_mag\_positive(self): # 6  
 *"""Test a list of positive values."""* argument = [4, 2, 3, 6]  
 expected = [2, 3, 4, 6]  
 ibm.list\_by\_mag(argument)  
 self.assertEqual(expected, argument, "The list contains only positive values.")  
  
 def test\_running\_list\_mag\_mix(self): # 7  
 *"""Test a list containing mixture of negative values, zeros and positive values."""* argument = [-2, -1, 2, 3, -5, 4]  
 expected = [-1, -2, 2, 3, 4, -5]  
 ibm.list\_by\_mag(argument)  
 self.assertEqual(expected, argument, "The list contains a mixture of negative values, zeros and positive values"  
 ".")  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 unittest.main()