Lab 4.4 – Insertion Sort with both an array list and a linked list

Learning Goals

1) Develop your understanding of the separation of interface and implementation.
2) Develop your skills in manipulating data in array lists and in linked lists.

Background

We previously implemented “intlist” using both an array and a linked list. In this assignment, we build upon both of these implementations. So far, we have implemented the following behaviors:

- ilAppend
- ilCopy (we only implemented for arrays)
- ilCreate
- ilDelete
- ilFind
- ilGet
- ilInsert
- ilSize

Your Task

In this assignment, you will implement 3 new behaviors for our existing intlist. You will implement each behavior in both the **array list** and **linked list** versions of the program.

These behaviors include the following:

- **ilPrint**: Prints the contents of a list in order.
- **ilInsertSorted:** Inserts a new element into a list in the correct position, assuming the current list is sorted in ascending order.
- **ilSort:** Creates a copy of an unsorted list and returns the new copy in sorted order. This method uses the insertion sort method of sorting to create the new list, by inserting every element into the new list via calls to ilInsertSorted for each element.

In this assignment, you have been given four files:

- **intlist.h:** New behaviors have already been added for you.
- **listtest.c**: Completely implemented test program for the new functionality. Use this test program for both the array list and linked list implementations.
- **arraylist.c** and **linkedlist.c** Correctly implemented code from class and prior assignments is provided for you. Your task is to add the three new functions ilPrint, ilInsertSorted, and ilSort.

Testing

You will need to separately test your implementations of **arraylist.c** and **linkedlist.c**

When testing the **linked list**, you would compile with a command like:

```
gcc -o test linkedlist.c listtest.c
```

When testing the **array list**, you would compile with a command like:

```
gcc -o test arraylist.c listtest.c
```

In both cases, the output of the programs should be the following:

```
$ ./test


Test 1 -- Should print 6, 12, 2, 19, -7:
0: 6
1: 12
2: 2
3: 19
4: -7


Test 2 -- Should print -7, 2, 6, 12, 19
0: -7
1: 2
2: 6
3: 12
4: 19


Test 3 -- Should print -7, 2, 6, 12, 19
0: -7
1: 2
2: 6
3: 12
4: 19
```