CS1 Spring 2019

Assignment 8: Linked List Insertion Sort

This is similar to the linear insertion sort you have already written, but uses a linked list. Here is the method:

1. Ask the user for the name of a file containing data. If it does not exist, the program should display an error, then ask for a new file name. Entering an asterisk (\*) should exit the program.
2. Create a “head” that is a pointer to your first list node and initialize it.
3. The way linear insertion works is this:
4. If the list is empty, create a new node with the number as the data and point the head to it.
5. If the list is not empty, search to find the two elements where the previous one is smaller than the one you have and the next one is larger (assuming an ascending-order sort.) This is your insertion point. Insert your new node into the list, as described in the textbook. If none of the elements is smaller, put the new item at the end. If it is smaller than the first element, the “head” must point to it.
6. Note that you will have to dynamically allocate each node, since you have no idea how many nodes you’ll need.
7. Once the list is sorted, print it by traversing the list and displaying each number, one per line. When you have printed all of the numbers, print the number of items in the list.
8. Once the list has printed, delete all of the items in it.
9. Go back to step 1 and ask for another file name.

There are two files on eLearning. Download these and test your program with them. You can assume that the files do not contain bad data. They contain only valid floating-point numbers.

|  |  |
| --- | --- |
| **Grading Criteria** | |
| Program behaves according to the specification, sorting correctly, requesting file names, and catching “file not found” errors. | 60% |
| Program is structured well, with functions to do most of the work rather than everything in the main loop. | 30% |
| Program documentation | 10% |

Additional grading guidelines:

1. Uses the linked list class in the STL: -60
2. Does not use a linked list: -60
3. Reads all of the numbers into an array or other structure, then does the list insertion: -30
4. Does not handle multiple files correctly: -20
5. Does not use dynamic allocation for list nodes: -30