Instructions for Module 14, Exercise #2 (add sorted)

In this exercise, you will fill in the details for a linked list by adding a function named **addSorted**, which inserts an item into a linked list so that it is always in sorted order.

First, download the **SortedList.cpp (https://miracosta.instructure.com/courses/31330/files/7025579/download?wrap=1)** ⇩ **(https://miracosta.instructure.com/courses/31330/files/7025579/download?download_frd=1)** program into a C++ project. The **main** function in this program is already completed, and is designed to add students randomly into a linked list in sort by name.

The **main** function calls another named **show** which takes a list of Student names and displays them on the console. The prototype for the **show** function has already been established, but the body of the **show** function (at the bottom of the code) should be empty. Find the heading for the **show** function and insert the following code. (Code circled in *red dashes* cannot be cut-and-pasted):

```
voiddshow(LinkedNodee*ptr)){{
    whilee(ptrr!==nullptr)){{
        couttt<<<ptr->nameee<<<" "";;
        ptrr==ptr->nextt;;
    }}
    couttt<<<endll;;
}}
```

This code traverses the nodes in the list beginning at the node referenced by the **ptr** variable. It prints the name of student in the node , then process all trailing nodes until the end of the list is reached. Normally it would be called so that **ptr** points to the first node in a list, but that is not a requirement.

Make sure that your code compiles, though it still won't work.

Next we need to complete the **addSorted** function (currently empty).

Add the code to process insertions at the beginning of the list (or if the list is empty). Inside the **addSorted** function, add the following code:

```
  iff(startt===nullptrr|||(start->namee>>name_to_add))){{
      startt==newwLinkedNode(name_to_add,,start));;
      returnnstartt;;       // new head of the list
  }}
```

Compile the program. It should run, but will only show a few names of the list, not the complete list.

Next let's deal with the situation where the name to be inserted into the list occurs between the beginning and the end of the list, or where the name to be inserted should be at the end of the list.

Under the code just added above, insert the following code. Notice that it uses two pointers into the list, one which will point to the node which comes *after* the node to be inserted, and one which points to the node *before* the node to be inserted:

```
  LinkedNodee*afterr==startt;;       // after the new node
  LinkedNodee*beforee==nullptrr;;    // before the new node
  whilee(afterr!==nullptrr&&&after->namee<==name_to_add)){{
      beforee==afterr;;
      afterr==after->nextt;;
  }}
  before->nextt==newwLinkedNode(name_to_add,,after));;
  returnnstartt;;                    // unchanged
```

Compile and run. The complete list of names should now print in sorted order!