ListLab1, extended

Recall that we introduced a basic ListNode class. Assume that the ListNode class also has a constructor that takes two appropriate parameters and accessor/mutator methods for each field.

Exercises

1. Draw a picture of the linked list constructed by the following commands.

```
    ListNode p = new ListNode("apple", null);
    ListNode a = new ListNode("banana", p);
```

2. Implement the three methods below.

```
    // returns a new node that is a copy of the argument node.
    ListNode* ListNode::copyNode(ListNode* arg)
    {






    }

    // returns a new list that is a copy of the original list.
    // this method is recursive!
    // example call:  ListNode* head2 = head->copyList(head);
    ListNode* ListNode::copyList(ListNode* arg)
    {





    }

    //returns a new linked list containing copies of each node in
    //the original list except the first node, maintaining the
    //order of the original list.  It is not correct to just
    //return a pointer to the 2nd node of the original linked
    //list.  This method is recursive.
    ListNode* ListNode::rest(ListNode* h)
    {









    }
```

3. Write the following six methods that operate on a linked list of ListNode objects. You may wish to make use of the three methods above.

// returns the value of the first node, or `NULL` if the list is empty
```
string first(ListNode* head)
```

// returns the value of the second node, or `NULL` if the list is empty or if there is only one node.
// hint: `second` could call the `first` of `rest`.
```
string second(ListNode* head)
```

//returns a reference to the last node in the list, or `NULL` if the list is empty.
```
ListNode* pointerToLast(ListNode* head)
```

//returns a copy of the last node (not just its value!). `copyofLast` can be recursive.
```
ListNode* copyOfLast(ListNode* head)
```

//returns a reference to a list whose first node's value is specified by the argument, and the
//first node's `next` links to the original list.
```
ListNode* insertFirst(ListNode* head, string arg)
```

//returns a reference to a list whose last node's value is specified by the argument, such
//that this last node has been appended to the original list and had its `next` is set to `NULL`
```
ListNode* insertLast(ListNode* head, string arg)
```

Assignment

Open ListLab1 and modify it to include all methods above. Specifically, construct a linked list and display the value of its first, second, pointer to last, and copy of the last node. Then prompt the user for a new value to be inserted and insert that value to both the front and end of the list. Print out the new list, which should have two more nodes than the original.

Sample run:
```
[computer, science, java, coffee, nonsense, boo, foo, hello]
First = computer
Second = science
Pointer to Last = hello at <address 1>
Copy of Last = hello at <address 2>
Insert what? what?
[what?, computer, science, java, coffee, nonsense, boo, foo, hello, what?]
```