LinkedList.java

```
package hwk2;

/**
* Linked List is a collection of data nodes.  All methods here relate to
* how one can manipulate those nodes.
*
* @author (your name)
* @version (a version number or a date)
*/
public class LinkedList
{
   private int length;          // number of nodes
   private ListNode firstNode;  // pointer to first node

   public LinkedList()
   {
       length=0;
       firstNode=null;
   }

   /** insert new String at linked list's head
    *
    * @param newData the String to be inserted
    */
   public void insertAtHead(String newData)
   {
       ListNode newnode = new ListNode(newData);
       newnode.next=firstNode;
       firstNode=newnode;
       length++;
   }

   /** remove and return data at the head of the list
    *
    *  @return the String the deleted node contains.  Returns null if list
empty.
    */
   public String removeHead()
```

```
{
    // implement me (AFTER writing some tests)!
    if (isEmpty())
    {
        return null;
    }
    else
    {
        String first=firstNode.data;
        firstNode=firstNode.next;
        length--;
        return first;
    }
}

/** insert data at end of list
 *
 * @param newData new String to be inserted
 */
public void insertAtTail(String newData)
{
    ListNode newnode = new ListNode(newData);
    if (isEmpty())
    {
        firstNode=newnode;
    }
    else
    {
        ListNode runner=firstNode;
        while (runner.next!=null) runner=runner.next;
        runner.next=newnode;
    }
    length++;
}

/**
 * search for first occurrence of value and return index where found
 *
 * @param value string to search for
```

```
    * @return index where string occurs (first node is index 0).  Return
-1 if value not found.
    */
   public int indexOf(String value)
   {
       ListNode runner=firstNode;
       int index=0;
       while (runner!=null)
       {
           if (runner.data.equals(value))  return index;
           runner=runner.next;
           index++;
       }
       return -1;  //not found
   }

   /**
    *  @return return linked list as printable string
    */
   public String toString()
   {
       String toReturn="(";
       ListNode runner=firstNode;
       while (runner!=null)
       {
           toReturn = toReturn + runner;  //call node's toString
automatically
           runner=runner.next;
           if (runner!=null)
           {
               toReturn = toReturn + ",";
           }
       }
       toReturn = toReturn + ")";
       return toReturn;
   }

   /**
    *
    * @return length of LL
```

```
    */
   public int getLength() {return length;}

   /**
    *
    * @return true if LL empty or false if not
    */
   public boolean isEmpty() {return getLength()==0;}
}
```

==========

ListNode.java:

```
package hwk2;

/**
* ListNode is a building block for a linked list of data items
*
* @author C. Fernandes
* @version 4/21/22
*/
public class ListNode
{
   public String data;
   public ListNode next;

   /** Non-default constructor
    *
    * @param value data for this node
    */
   public ListNode(String value)
   {
       data = value;
       next = null;
   }

   /**
    * returns data as a printable string
    */
```

```
    public String toString()
    {
        return data;
    }
}
```