* + **Some common coding tips for handling linked lists in action (remember LLNode)**
    - **Solutions for linked list depend on traversing the list, and possibly depending on the location desired (at end, throughout/middle, beginning) taking a slightly different approach to designing a solution either iteratively or recursively.**
      * **Some iterative solutions are better, e.g. printing a linked list using recursion involves unnecessary overhead**
      * **But at times if/when a recursive solution exists and iterative is overly complicated**
    - **Solution for printing linked list using recursion:**
      * **TBD INSERT HERE**
      * **if(listRef != null) checks for the base case when listRef is null it well end**
      * **general cases is inside conditional uses reference to getLink (smaller caller)**
    - **Solution for traversing and adding element to LL**

**LLNode<String> recInsertEnd(String newInfo, LLNode<String> listRef)**

**// Adds newInfo to the end of the listRef linked list**

**{**

**if (listRef != null)**

**listRef.setLink(recInsertEnd(newInfo, listRef.getLink()));**

**else**

**listRef = new LLNode<String>(newInfo);**

**return listRef;**

**}**

* + - * + **NOTE: it is tempting to use if(listRef.getLink() != null) but that doesn’t work because of the problem created by having an empty list**
* **An iterative solution to problems that may be defined recursively is sometimes better!**
  + **Iteration works as a good replacement in the case of tail recursion, recursive call is last statement in the method.**
    - **E.g. Iteratively Printing Linked List is better than recursion**
      * **while(listRef != null) is a more commonplace statement to provide iteration**
      * **inside the body of loop listRef reassigned to remaining portion of linked list** 
        + **That “link” list is acessible through getLink**
  + **In other cases it can’t be done, but a stack implementation in the system can function as a data structure to store the values thus mimicking the recursion in the method**
    - **E.g. Print in reverse order using a LinkedStack implementation!**

Recursion vs. Iteration

Examples of

//placeholder for recursive function:

static void recRevPrintList(LLNode<String> listRef)

**static** **void** iterRevPrintList(LLNode<String> listNode)

{

//checks for empty list

**if**(listNode != **null**){

//use a stack

LinkedStack<String> temp = **new** LinkedStack<String>();

//traverses the list reference and pushes to the stack from list

**while** (listNode!= **null**) {

temp.push(listNode.getInfo());

listNode=listNode.getLink();

}

//finally uses stack to print the reverse order

**while** (!temp.isEmpty()){

System.*out*.println(" " + temp.top());

temp.pop();

}

}

}