CSCI 2170
Recursion with linked list

Example 1:

```
//Write the content of a linked list using recursion
// StringPtr is a pointer to a linked list of characters
void WriteString(nodePtr StringPtr)

{
  if (StringPtr != NULL)
  {  // write the first character
     cout << StringPtr→Item;


     // write the string minus its first character
     WriteString(StringPtr→Next);
  }  // end if
}  // end WriteString
```

Example 2:

```
//Write the content of a linked list of items backwards
void WriteBackward(nodePtr StringPtr)

{
  if (StringPtr != NULL)
  {
     // write the string minus its first character backward
     WriteBackward(StringPtr→Next);

     // write the first character
     cout << StringPtr→Item;
  }  // end if
}  // end WriteBackward
```

Example 3:

```
// insert an item into a sorted linked list
//  this example assumes the list is built in a program without a listclass
void LinkedListInsert(nodePtr& HeadPtr, itemType NewItem, bool& Success)
{
  if ((HeadPtr == NULL) || (NewItem < HeadPtr→item))
  {  // base case: insert NewItem at beginning of the linked list to which HeadPtr points
     ptrType NewPtr = new node;
     Success = bool(NewPtr != NULL);
     if (Success)
     {
        NewPtr→Item = NewItem;
        NewPtr→Next = HeadPtr;
        HeadPtr = NewPtr;
     }  // end if
```

```
  }
  else // move down the list by making recursive calls to the next node in the list
    LinkedListInsert(HeadPtr→next, NewItem, Success);

}  // end LinkedListInsert
```

Recursive function as a member of listClass
!! head pointer is private data in listClass !!

listclass.h

```
class ListClass
{
public:
        …
        void ListInsert(listItemType newItem, bool & Success);
       …
private:
        …
        nodePtr  Head;
        void LinkedListInsert(nodePtr& HeadPtr, ListItemType NewItem, bool& Success);
};
```

listclass.cpp

```
void ListClass::ListInsert (ListItemType newItem, bool & Success)
{
        RecursiveInsert(Head, newItem, Success);
}
// same implementation as above
void ListClass::RecursiveInsert(nodePtr& HeadPtr, ListItemType NewItem, bool& Success)
{
  if ((HeadPtr == NULL) || (NewItem < HeadPtr→item))
  {  // base case: insert NewItem at beginning of the linked list to which HeadPtr points
    ptrType NewPtr = new node;
    Success = bool(NewPtr != NULL);
    if (Success)
    {
      NewPtr→item = NewItem;
      NewPtr→next = HeadPtr;
      HeadPtr = NewPtr;
    }  // end if
  }
  else
    RecursiveInsert(HeadPtr→next, NewItem, Success);

}  // end LinkedListInsert
```

How to write ListDelete and RecursiveListDelete?
How about ListRetrieve?