CS 3345 HON Homework 1

1. Write the function void insertAtTail(int ky). Don't add any class variables to the List class

```
void insertAtTail(int ky) { // inserts at the end of the list
      if(head == null)
        head = new Node(head, ky);
      else {
        Node temp = head;
        while(temp.getNext() != null) {
          temp = temp.getNext();
          System.out.print("debug");
        }
        temp.putNext(new Node(temp, ky));
      }
    }
```

2. Write the private iterative function void delete(int ky) using only ONE reference variable that marches along the list (my notes use two reference variables, ref and prev).

```
void delete(int ky) { // delete the element or do nothing if ky doesn't exist
      if(head != null) {
        if(head.getKey() == ky)
          head = head.getNext();  // remove first element
        else {
          Node ref = new Node(head, 0);
          while(ref != null && ref.getKey() != ky) {
            ref = ref.getNext();
          }
          if(ref != null)
            ref.putNext(ref.getNext());
        }
      }
```

3. Write the private recursive function int maxElement(Node x)

```
private int maxElement(Node x) {
      if(x != null) { // return -1 if list is empty
        if(x.getNext() == null) // end of list
          return x.getKey();
        else {
          int max = maxElement(x.getNext());
          if(max < x.getKey())
            return x.getKey();
          return max;
        }
      }
      return -1;
    }
```

4. Write the private recursive function int sum(Node x) to find the sum of the keys stored in a List.

```
private int sum(Node x) {
      if(x != null) // return -1 if list is empty
      {
        if(x.getNext() == null) // end of list
          return x.getKey();
        return x.getKey() + sum(x.getNext());
      }
      return -1;
    }
```

5. Write the private recursive function int length(Node x) to find the number of keys in a List.

```
private int length(Node x) {
      if(x != null) { // return 0 if list is empty
```

```
        if(x.getNext() == null)
          return 1;
        else return 1 + length(x.getNext());
      }
      return 0;
    }
```

6. Assume the addition of two recursiveDelete fuctions, one public and one private. Write both functions.
7.

TA(n) has run time *O(n^2)* and *TB(n)* has run time 8(O(n^3))*

TA would be faster for large values of *n*

8.
9.
10.