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();
        }
        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.

```
public void recursiveDelete(int ky) {
        recursiveDelete(ky, head);
    }

public Node recursiveDelete(int ky, Node n) {
        if (n.getKey() == ky)
          return n.getNext();
        return new Node(n.getKey(), n.getNext().remove(ky, n));
    }
```

7. Algorithm A has running time *TA(n) = 106 + 104 × n + 105 × n2* and algorithm B has running time *TB(n) = 3 × n3*, where n is the number of values the algorithms processes. Give the "big O" equations for the running times and state which algorithm is fastest for large *n*.

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

Algorithm A would be faster for large values of *n*.

8. Algorithm C has running time *TC (n) = O(n)*, algorithm D has running time *TD(n) = O(log n)*, and algorithm E has running time *TE(n) = O(p(n))*. Which algorithm is the fastest and which is the slowest for large *n*?

For large values of *n*, C is the slowest and D is the fastest.

9. A linked list as defined above holds N nodes. What is the runtime in "Big O" notation of an algorithm that searches the list for a given key when:

   (a) The list elements are arbitrarily ordered > *O(n)*

   (b) The list elements are arranged in increasing order > *O(n) because list is singly-linked*

10.

(a)

(b)