Practice Problems for Recursion
1. Write the recursive function
   
   ```java
   int Sum( ArrayList<Integer> L, int i)
   ```
   
   that returns the sum of the elements of L at index n or higher. The sum of the entire list will be `Sum(L, 0)`. Yes, you can do this just as easily with a loop, but do it recursively for the practice.
2. Write the recursive function
   
   ```java
   int Largest( ArrayList<Integer> L, int i)
   ```
   that returns the largest element of L at index i or higher.
3. Write a recursive function that reverses a string:
   String reverse(String s)
4. Write a recursive function to determine if a string is a palindrome (i.e. if it is equal to its reverse, such as “bob”)
5. Implement BinarySearch recursively. You have a sorted array int A[ ]; you need to write

   boolean Search(int A[ ], int lowIndex, int hiIndex, int x)

that returns true if x is one of the elements of A between the two indices, and false if it isn’t. X is an element of A if Search(S,0,A.length-1, x) returns true.
Here is a Node type for a binary search tree that holds integer data:

```java
class Node {
    int data;
    Node leftChild, rightChild;
}
```

Give a recursive function

```java
ArrayList<Integer> inOrder(Node p)
```

that returns an inOrder traversal of the tree rooted at Node p.

Note that if L and M are ArrayLists then L.addAll(M) adds all of the elements of M onto L.