1. Recall the recursive function $C(n, k)$ in the class BinomialCoefficients discussed in lecture and posted on the webpage. Write a box trace of the function call $C(5, 3)$. Use this trace to find the value of $C(5, 3)$. Notice that in the full recursion tree for $C(5, 3)$, the value $C(3, 2)$ is evaluated 2 times, and $C(2, 1)$ is evaluated 3 times. Suggest a modification to the function that would allow it to avoid computing the same values multiple times. Carry out your suggestion by writing the Java code.

2. Write a recursive function called $\text{sum}(n)$ that computes the sum of the integers from 1 to $n$. Hint: emulate the factorial function discussed in lecture (replace multiplication by addition). Modify your answer to recursively compute the sum of the integers from $n$ to $m$, where $n \leq m$ (return 0 if $n > m$).

3. Write recursive functions that determine the sum of the elements in an int array. Do this in 3 ways.
   a. Write a recursive function $\text{sumArray1}(\text{int}\ [\text{A}], \text{int}\ n)$ that returns the sum of the leftmost $n$ elements of $\text{A}[]$. Get the $n^{th}$ element from the left, compute the sum of the leftmost $(n - 1)$ elements recursively, then return the sum.
   b. Write a recursive function $\text{sumArray2}(\text{int}\ [\text{A}], \text{int}\ n)$ that returns the sum of the rightmost $n$ elements of $\text{A}[]$. Get the $n^{th}$ element from the right, compute the sum of the rightmost $(n - 1)$ elements recursively, then return the sum.
   c. Write a recursive function $\text{sumArray3}(\text{int}\ [\text{A}], \text{int}\ p, \text{int}\ r)$ that returns the sum of the subarray $\text{A}[p...r]$. Hint: use MergeSort() as a model.

4. Write a modification of the recursive function $\text{BinarySearch()}$ that prints out the sequence of array elements that are compared to the target.

5. What output does the following program produce?

```java
public class problem5 {
    public static int getValue(int a, int b, int n){
        int x, c;
        System.out.println("arrive: a = " + a + " b = " + b);
        c = (a+b)/2;
        if( c*c <= n ){
            x = c;
        }else{
            x = getValue(a, c-1, n);
        }
        System.out.println("depart: a = " + a + " b = " + b);
        return x;
    }

    public static void main(String[] args){
        System.out.println(getValue(3, 13, 5));
    }
}
```
6. The following Java method converts a positive decimal integer to base 8 (octal) and displays the result. Explain how the function works and trace it on the input \( n=100 \).

```java
static void displayOctal(int n){
    if(n>0){
        if(n/8>0){
            displayOctal(n/8);
        }
        System.out.println(n%8);
    }
}
```

7. Use what you learned in problem 6 above to create a recursive function called `integerToString()` that returns a String representation of an integer \( n \) expressed in base \( b \). For instance the function call `integerToString(100, 8)` would return the String “144”, which is what was printed in problem 6.

```java
static String integerToString(int n, int b){
    // your code starts here

    // your code ends here
}
```

8. Recall the IntegerList ADT discussed in class whose states were the finite integer sequences, and whose operations were `isEmpty()`, `size()`, `get()`, `add()`, `remove()`, and `removeAll()`. Write the methods described below using only these six ADT operations. In other words you are writing methods belonging to a client of `IntegerList`.
   a. Write a static void method called `swap(IntegerList L, int i, int j)` that will interchange the items currently at positions \( i \) and \( j \) of the List.
   b. Write a static int method called `search(IntegerList L, int x)` that will perform a linear search of \( L \) for the target \( x \). `search()` will return the List index where \( x \) was found, or it will return 0 if no such index exists. (Recall List indices range from 1 to `size()`.)
   c. Write a static void method called `reverse(IntegerList L)` that reverses the order of the items in \( L \).
9. Given classes Node and NodeTest defined below, answer the following questions.
   a. Draw a picture of the linked data structure at point (a) in function main() of NodeTest.java.
   b. Trace execution of main() up to point (b) and write the output as it would appear on the screen.
   c. Write instructions that will insert a new Node with item value 4 into position 3 of the list, i.e. insert the 
      new Node between the 7 and the 5.

   // file: Node.java
   public class Node{
       // fields
       public int item;
       public Node next;
       // constructor
       public Node(int x){
           item = x;
           next = null;
       }
   }

   // file: NodeTest.java
   public class NodeTest{
       public static void main(String[] args){
           Node H = new Node(9);
           H.next = new Node(7);
           H.next.next = new Node(5);
           // part (a) refers to this point in the code
           for(Node N=H; N!=null; N=N.next) System.out.print(N.item+" ");
           System.out.println();
           // part (b) refers to this point in the code
           // part (c) refers to this point in the code
           // your code goes here
           // your code ends here
   }
}
10. Given the Node class in problem 8 above and a linked list based on that class, fill in the function definitions below.

a. Write a recursive function called `printForward()` that prints out the items from head to tail.

```java
static void printForward(Node H) {
    // your code starts here
    // your code ends here
}
```

b. Write a recursive function called `printBackward()` that prints out the items from tail to head.

```java
static void printBackward(Node H) {
    // your code starts here
    // your code ends here
}
```