CMPS 12B
Introduction to Data Structures
Quiz 1 Practice Problems

1. Recall the recursive function `C(n, k)` in the class `BinomialCofficients` 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. (Don't write the code, just explain it in words.)

2. Write a recursive function called `sum(n)` that computes the sum of the integers from 1 to *n*. Hint: recall the recursive function `fact(n)` in the class `Factorial` discussed in lecture and posted on the webpage.

3. Write a recursive Java method that takes two non-negative integers *n* and *m* as input, then returns the sum of the integers from *n* to *m* (inclusive) if $n \leq m$, and returns 0 if $n > m$. Do this in two ways as described below.

   a. Determine the sum of integers from n to $m - 1$ recursively, then add *m* to the result.
   b. Split the sequence of integers from *n* to *m* (roughly) in half, recur on the two half-sequences, then add the results. Hint: model this function on `mergeSort()`.

4. Write a recursive function called `sumArray()` that determines the sum of the integers in an array `A[0...n-1]`. Do this in 3 ways.
   a. Recur on `A[0...n-2]`, add the result to `A[n-1]`, then return the sum.
   b. Recur on `A[1...n-1]`, add the result to `A[0]`, then return the sum.
   c. Split `A[0...n-1]` into two subarrays of length (approximately) `n/2`, recur on the two subarrays, add the results and return the sum. Hint: think about `MergeSort()`.

5. Write a modification of the recursive function `BinarySearch()` that prints out the sequence of array elements that are compared to the target.

6. What output does the following program produce?

```
public class problem6 {
   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));
   }
}
```

7. 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.

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

8. 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.

9. The Fibonacci function $F(n)$ is defined by $F(0)=0$, $F(1)=1$ and the recurrence formula $F(n)=F(n-1)+F(n-2)$ for $n \geq 2$. Write a *recursive* Java method that calculates $F(n)$.

10. Perform a box trace of the function call `F(5).` for the Fibonacci function from the previous problem. Each box should state the value of *n* for the recursive call that it represents. What integer is returned?