1. Write a recursive method that takes as input a string $S$ and returns the reversed version of $S$. For example, `reverse("abcd")` returns "dcab". You will probably find the `substring(int beginIndex)` method of the `String` class to be useful. When you have finished, demonstrate your implementation to the instructor.

2. Write a recursive method to implement binary search on a sorted array. The input to the binary search method is a sorted array $M$ of integers along with the integer $k$ that we are searching for. If $k$ is in $M$, the method should return the position in $M$ where $k$ is located. Otherwise, if $k$ is not in $M$, the method should return $-1$.

Binary search works as follows. In each step, the algorithm compares $k$ with the integer, $mid$, in the middle of the array. If $mid = k$, then $k$ has been found in the array, so its position is returned. Otherwise, if $k < mid$, then the algorithm recursively searches the sub-array to the left of $mid$ or, if $k > mid$, the algorithm recursively searches the sub-array to the right. If the remaining array to be searched contains 0 elements, then $k$ cannot be found in the array, so $-1$ is returned.

Instead of explicitly creating new sub-arrays, you may use two integer parameters `first` and `last` to define a sub-array of $M$. If you choose to do this, your method should look like:

```java
public static int binarySearch(int[] m, int k, int first, int last)
```

HINT: There are two base cases (non-recursive cases).

When finished, demonstrate your binary search method to the instructor.

3. The following is the recurrence relation that defines Fibonacci numbers. $Fib(n) =$

   (a) $n$ if $n = 0$ or $n = 1$
   (b) $fib(n - 1) + fib(n - 2)$ if $n \geq 2$

First, implement a recursive method `fib(int n)` that takes as input an integer $n$ and returns the corresponding Fibonacci number. Then write code to count the number of recursive calls that occur when calling `fib(int n)`. Run your code with different values of $n$, and observe the number of recursive calls. When finished, please demonstrate your `fib` method, and present your observations to the instructor.

4. First, implement a recursive method `factorial(int n)` that returns $n!$. Then, as in the previous exercise, write code to count the number of recursive calls that occur in calculating $n!$. Perform experiments for different values of $n$. 

CS401 Lab Nov. 6
Compare your results to those of the fib method in the previous exercise. When finished, please demonstrate your factorial method, and present your observations/conclusions to the instructor.