Static Methods - *Combinatorics* FRQ

Assume that the classes listed in the Java Quick Reference have been imported where appropriate.

Unless otherwise noted in the question, assume that parameters in method calls are not null and that methods are called only when their preconditions are satisfied.

In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods will not receive full credit.

This question involves computing factorials and using factorials to compute the number of possible ways that items can be selected from a group of choices. You will write two methods in the *Combinatorics* class that follows.

```
public class Combinatorics
{
   /** Precondition: n is between 1 and 12, inclusive.
    * Returns the factorial of n, as described in part (a).
   */
   public static int factorial(int n)
   { /* to be implemented in part (a) */ }

   /** Precondition: n and r are between 1 and 12, inclusive.
    * Determines the number of ways r items can be selected from
      n choices and prints the result, as described in part (b).
   */
   public static void numCombinations(int n, int r)
   { /* to be implemented in part (b) */ }
}
```

(a) In mathematics, the factorial of a positive integer *n*, denoted as *n!* , is the product of all positive integers less than or equal to *n*.

The factorial of *n* can be computed using the following rules.

Case I: If *n* is 1, then the factorial of *n* is 1.

Case II: If n is greater than 1, then the factorial of *n* is equal to *n* times the factorial of *(n - 1)*.

The factorial method returns the factorial of *n*, as determined by case I and case II. Write the factorial method below.

```
/** Precondition: n is between 1 and 12, inclusive.
 * Returns the factorial of n, as described in part (a).
 */
public static int factorial(int n)
```

(b) A combination is a selection of items from a group of choices when the order that the items are selected does not matter. For example, if there are four available choices (A, B, C, and D), there are six different ways that two items can be selected (A and B, A and C, A and D, B and C, B and D, C and D). The number of possible combinations of *r* items from a group of *n* choices can be calculated according to the following rules.

If *r* is greater than *n*, then the number of possible combinations is *0* .

If *r* is not greater than *n*, then the number of possible combinations is equal to *n!/(r!(n−r)!)*.

The *numCombinations* method is intended to calculate the number of possible combinations of *r* items from a group of *n* choices and print the result. Examples of the intended behavior of the method are shown in the table.

Method Call	**Message Printed**	**Explanation**
numCombinations(2, 4)	*There are 0 ways of choosing 4 items from 2 choices.*	There is no way to select 4 items from 2 choices since 4 is greater than 2.
numCombinations(5, 3)	*There are 10 ways of choosing 3 items from 5 choices.*	The number of possible ways to select 2 items from 5 choices is 5!/(3!(5−3)!) , or 5!/((3!)(2!)) , which evaluates to 10.

Write the *numCombinations* method below. Assume that factorial works as specified, regardless of what you wrote in part *(a)*. You must use factorial appropriately to receive full credit.

```
/** Precondition: n and r are between 1 and 12, inclusive.
 * Determines the number of ways r items can be selected
 * from n choices and prints the result, as described in part (b).
*/
public static void numCombinations(int n, int r)
```