Test 4

The fourth test will be 75 minutes, will consist of two parts and will take place next week.

The programming part will be about Chapter 2-6, excluding Section 2.6, 4.5, and 6.8.8.

The "written" part will be about Chapter 2-6, excluding Section 2.6, 4.5, and 6.8.8.

During the test, you will have access to the textbook. You may bring a blank piece of paper to the test.

Chapter 7: Recursion

EECS 1030

moodle.yorku.ca

Print stars

```
public class Print
{
  /**
   * Print the specified number of *s.
   *
   * @param n the number of stars.
   * @pre. n >= 0
   */
  public static void stars(int n)
  {
    ...
  }
}
```

Print stars

```
public class Print
{
  /**
   * Print the specified number of *s followed by the
   * same number of +s.
   *
   * @param n the number of stars and plusses.
   * @pre. n >= 0
   */
  public static void starsAndPlusses(int n)
  {
    ...
  }
}
```

Correctness

- Prove the base case(s) correct.
- Assume the recursive call(s) to be correct. Prove the remaining case(s) correct.

Termination

- Define the size of each invocation of the recursive method, also known as the *variant expression*. The size has to be a natural number (that is, a non-negative integer).
- Prove that each recursive invocation has a smaller size than the original invocation.

Terminates?

```
public static void done(int n)
{
  if (n == 1)
  {
    System.out.println("done");
  }
  else
  {
    if (n % 2 == 0)
    {
       Print.done(n / 2);
    }
    else
    {
       Print.done(3 * n + 1);
    }
  }
```

Powers of 2

```
/**
 * Returns 2 raised to the given exponent.
 *
 * @param exponent the exponent.
 * @pre. exponent >= 0
 * @return 2 raised to the given exponent.
 */
public static long power(int exponent)
{
   ...
}
```

Fibonacci

```
/**
 * Returns the nth Fibonacci number.
 *
 * @param n a number.
 * @pre. n >= 1
 * @return the nth Fibonacci number.
 */
public static long fibonacci(int n)
{
   ...
}
```

Zeroes

```
/**
 * Returns the number of zeroes in the given number.
 *
 * @param n a number.
 * @pre. n >= 0
 * @return the number of zeroes in the given number.
 */
public static long zeroes(int n)
{
   ...
}
```

Convert to binary

```
/**
 * Prints the binary representation of the given number.
 *
 * @param n a number.
 * @pre. n >= 0
 */
public static void binary(int n)
{
   ...
}
```