Recursion

1. **GreatestCommonDivisor.java**: One of the most famous algorithms known was stated over two thousand years ago by the Greek mathematician Euclid. *Euclid's algorithm* provides a method of finding the greatest common divisor (gcd) of a pair of natural numbers. The algorithm is based on the following properties of gcd:
   Rule 1: If m = n, then gcd(m, n) = m
   Rule 2: If m > n, then gcd(m, n) = gcd(n, m-n)
   Rule 3: If m < n, then gcd(m, n) = gcd(n, m)
   Write a recursive method based on the definition above.

2. **Square.java**: Write a method that implements this definition of *square numbers*:
   ```
   square(1) = 1
   square(N) = square(N-1) + 2N – 1
   ```

   Aside: where did this crazy definition of *square numbers* come from?
   Easy: this is from simple algebra:
   $(N-1)^2 = N^2 - 2N + 1$
   rearrange to get:
   $N^2 = (N-1)^2 + 2N - 1$

3. **PrimeNumber.java**: A *prime number* is an integer that cannot be divided by any integer other than one and itself. For example, 7 is a prime number because its only divisors are 1 and 7. The integer 8 is not a prime number because its divisors are 1, 2, 4 and 8.
   Another way to define prime is
   ```
   prime (N) = prime (N, N-1)
   prime (N, 1) = true
   prime (N, D) = false if D divides N, prime (N, D-1)
otherwise
   ```

   For example,
   ```
   prime (4) = prime (4, 3)
   prime (4, 3) = prime (4, 2)
   prime (4, 2) = false
   ```

   Another example,
   ```
   prime(7)   = prime(7,6)
   prime(7,6) = prime(7,5)
   prime(7,5) = prime(7,4)
   prime(7,4) = prime(7,3)
   prime(7,3) = prime(7,2)
   prime(7,1) = true
   ```

Translate the math-like definition of prime into two Java methods that return `boolean`. Use the `%` operator to test divisibility. Put your method into a class, write a testing class, and test your program.

4. **Power.java**: Write a recursive function for X to the power of N. Test it by prompting the user for the value of X and N, calling the function and printing out the result.