Modular Programming II:

1. Design a method called **computeFactorial** that receives a positive integer and returns the factorial for that number. If the number is negative the method should return None.

2. Design a method called **isLeapYear** that receives a number and returns False for common years and True for leap years. You may know that a year is considered to be a leap year if it is divisible by 4, unless it is also divisible by 100. A year is not a leap year if it is divisible by 100 unless it is also divisible by 400.

3. Design a method called **computeDaysInMonth** that returns the number of days for the month and year that are received as arguments. You may use the method leapYear above. If the values are not valid the method should return -1.

4. Design a method called **getDayOfWeek** that receives a list containing three integers (day, month and year) and returns the day of the week for that date (Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday).

   You can use the following algorithm to get a number between 0 (Sunday) and 6 (Saturday) corresponding to the day in the week for a given date:
   ```
   a = (14 - month) / 12
   y = year – a
   m = month + 12 * a – 2
   d = (day + y + y/4 - y/100 + y/400 + (31*m)/12) mod 7
   ```

5. Design a method called **powerIt** that receives two integers and raises the first number to the second. You may use the product or recursion and check the values. If no exponent is provided the first number is raised to 0.

6. Design a method called **getNumberOfDigits** that receives one number (it can be real, integer, positive or negative) and should return the number of digits it contains. If the parameter is not valid the method should return None. Extend this function to other numeric systems (hexadecimal, decimal, binary, octal).

7. Design a method called **isPrime** that receives one integer positive number greater than 0 as parameter. The method should return True if the number is prime or False if not prime. If the parameter is not valid the method should return None.

8. Design a method called **solveSecondOrderEquation** that receives three integer positive numbers corresponding to the coefficients of a second order equation ($ax^2+bx+c=0$) and computes its possible solutions. If the parameters are not valid the method should return None.

9. Design a method called **getPrimeDivisors** that receives a positive number as a parameter and returns a list containing its prime divisors. If the parameter is not valid the method should return None.

10. Design a method called **isFriendNumber** that receives two positive numbers and returns True if the numbers are friends, False otherwise. Two numbers are considered to be friends if the sum of its divisors, except the given number, is equal to the second and vice versa.