Exercise n. 1

Given the integers ***first***, ***last*** and ***numOfThreads***, implement the *PrimeCalculator* program, a multi-threaded Java program which computes how many prime numbers between ***first*** and ***last*** by using ***numOfThreads*** threads. A prime number is a natural number greater than 1 whose only factors are 1 and itself; in other words, a natural number is a prime number if it can be divided, without a remainder, only by itself and by 1.

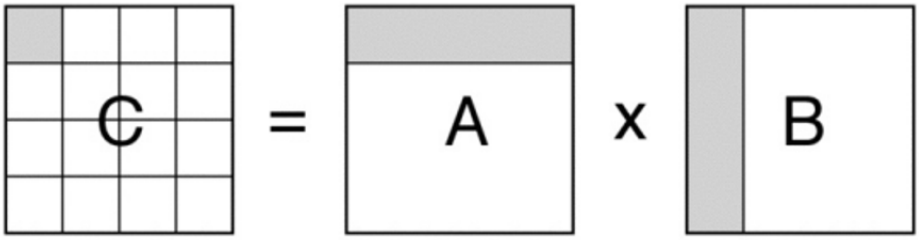
The multi-threaded program can be implemented by creating an array of threads, by splitting the initial range [*first, last*] in several sub-ranges and assigning to each thread (*PrimeCalculatorThread*) a specific sub-range to be computed. Implement the code, such that it works for generic values of ***first***, ***last*** and ***numOfThreads***.

The output of the program should be generated following the example reported below:

```
first = 1
last = 20
numOfThreads = 3

Thread-0 [1,6]  started!
Thread-1 [7,12]  started!
Thread-2 [13,20]  started!

Thread-0 [1,6]  completed!
Thread-1 [7,12]  completed!
Thread-2 [13,20]  completed!

Thread-0 [1,6]: prime numbers found = 3 (2,3,5)
Thread-1 [7,12] : prime numbers found = 2 (7,11)
Thread-2 [13,20] : prime numbers found = 3 (13,17,19)

Total number of prime numbers found = 8
```

Exercise n. 2

Given a ***n**x**m*** matrix of integers **A** and a ***m**x**k*** matrix of integers **B**, the student has to implement a multi-threaded Java program which computes the multiplication **n**x**k** matrix **C**, where **C = A x B**.



Remember that the generic element *C[i,j]* $(0 \leq i \leq n - 1,\ 0 \leq j \leq k - 1)$ is computed as follows

$$c_{ij} = \sum_{k=1}^{m} a_{ik} b_{kj}$$

The multi-threaded program can be implemented by creating a ***n**x**k*** matrix of threads, called *threadMatrix*. The generic **threadMatrix[i,j]** thread instance can process the **i^{th}-row of A** and the **j^{th}-column of B**, and it can compute the value **C[i,j]** $(\mathbf{0 \leq i \leq n\text{-}1,\ 0 \leq j \leq k\text{-}1})$. Implement the code, such that it works for generic values of ***n***, ***m***, ***k*** and ***numOfThreads***. The output of the program should print the matrices A, B and C, following the example reported below:

```
Matrix A
3       7
3       2
6       5
4       8

Matrix B
3       7       2
3       2       9

Matrix C
30      35      69
15      25      24
33      52      57
36      44      80
```

Exercise n. 3

Given a ***n**x**m*** matrix of integers **M** (***n*** rows and ***m*** columns), the student has to implement a multi-threaded Java program which computes the *saddle point* of the matrix.

A *saddle point* is an element of the matrix such that it is the ***minimum element in its row*** *and* ***maximum in its column***.

For example, the value 4 (element at row 2 and column 3) is a saddle point in the following matrix (it is the minimum in the 2nd-row and the maximum in the 3rd-column):

Matrix M

3	7	2	3	2
3	2	9	2	2
6	5	6	**4**	8
5	2	2	3	2

The task must be performed in concurrency, by splitting the original task in **n** row tasks and **m** column tasks, each one performed by a thread (**n** *RowThread* instances and **m** *ColumnThread* instances). Implement the code, such that it works for generic values of **n** and **m**.

The output of the program should be generated following the example reported below:

```
Matrix M
 3  7  2  3  2

 3  2  9  2  2

 6  5  6  4  8

 5  2  2  3  2


The saddle point is in M[2,3] = 4
```