Exercises 12

Sieve of Erathostenes

Exercise 1.12 - Price of bus tickets

In this exercise, we want to run a bus line that connects to Vierzon Orleans through Salbris Nouans, Lamotte-Beuvron and La Ferte Saint Aubin. Each section of the route has a price and the price of a ticket is the sum of the prices of the sections it contains. The line operates in both directions, the price is the same in both directions. In this exercise, we use only arrays

We will store in a array the names of cities, putting them in the order in which they are served (the order given above). In another array, we will store the prices of different trunks.

These prices are as follows:

départ	arrivée	prix
Vierzon	Salbris	3.20
Salbris	Nouans	1.80
Nouans	Lamotte-Beuvron	2.30
Lamotte-Beuvron	La Ferté Saint-Aubin	4.20
La Ferté Saint-Aubin	Orléans	5.00

Before starting to program, draw two pictures on paper and think about algorithms corresponding to different questions.

Question 12.1.1

1. **write** a method to find the index of a city whose name is given as a parameter (ie. its position in the first array).
2. **write** a method that calculates the price of a trip given the names of the cities of departure and arrival.

Question 12.1.2

We want to establish a sliding scale depending on the number of trunk covered on: the first trunk we paid full price, the second with 10% discount, the second with 20%, etc.. **Write** a method that performs the calculation of the price of a ride according to this principle.

Exercise of 12.2 – Sieve of Erathostenes

The method for calculating Erathostenes primes up to a given number **n**. Note: if the prime number is only divisible by 1 and itself.
The principle of sieve is to consider an array of all the numbers between 2 and **n** and successively remove all multiples of successive primes.
Take for example n = 15.
The table is as follows.
2 3 4 5 6 7 8 9 10 11 12 13 14 15

We begin by scratching all the multiples of 2 in the table. This gives:
2 3 **4**/ 5 **6**/ 7 **8**/ 9 **10**/ 11 **12**/ 13 **14**/ 15

It then seeks the next number not crossed out in the table. Is 3. It scratches all multiples of 3. This gives:
2 3 **4**/ 5 **6**/ 7 **8**/ **9**/ **10**/ 11 **12**/ 13 **14**/ **15**/

It then seeks the next number not scratched. This is 5. It lines the multiples of 5. Etc..
We can continue until we saw all the numbers in the array.
In fact, we can show that we can stop it before: just having deleted all the multiples of primes smaller than n, and the result is the same. (in our example, **√ 15 => 3** - so the last number to look is 3).

Question 12.2.1 - Simple version of sieve

Write a function that takes the number **n** as input parameter and returns the array of the sieve in the form of a boolean array where boolean where we note if the integer corresponding to a given index is scratched or not. Note that this array starts at 0 instead of starting with 2 as in the presented method. To simplify things, we suggest you simply ignore the first two cells of the table. By convention, 0 and 1 are not considered prime numbers.
Also note that the size of the returned array depends on the parameter **n**.
The square root function is written in Java as `Math.sqrt()`. It takes one parameter (`int`) and returns a result of type `double`.

Question 12.2.2 - Primality test

Write a function that checks if a number is prime. For this, use an Erathostenes sieve and see if that number is scratched or not in the table.

Question 12.2.3 - Sieve, sophisticated version

Write a function that computes all primes below a certain parameter **n**. The result will be given in the form of an integer array whose elements are primes. The calculation is the same as in question 1, but the form of the result is different.