Compte Rendu TP1
Larbaoui Riyad, Louis Maury

Exercice 1 :

Le résultat des tests :

```
riyad@MacBook-Pro-de-riyad tests-mocha % npx mocha exo1.spec.js


  ex. 1
    sum(...terms)
      ✔ should raise an error if not term to sum
      ✔ should sum 0 + 0 = 0
      ✔ should sum 1 term
      ✔ should sum 2 terms
      ✔ should sum 3 terms


  5 passing (8ms)
```

L'exercice 1 consistait à prendre plusieurs valeurs en entrée et à en faire la somme, pour prendre plusieurs valeurs en entrée, nous avons utilisé l'operateur "..." qui est un opérateur qui permet de collecter plusieurs éléments

Exercice 2 :

```
riyad@MacBook-Pro-de-riyad tests-mocha % npx mocha exo2.spec.js


  ex. 2
    filter(array, predicate)
      ✔ should return a new array
      ✔ should keep only items for those the predicate is true
      ✔ should not mutate the original array


  3 passing (28ms)
```

Pour cette exercice 2 nous avons coder une fonction de filtre, pour cela nous prenons une

fonction en entrée qui va nous permettre de filtrer (callback) et un tableau sur lequel nous allons appliquer cette opération de filtre

Exercice 3:

```
riyad@MacBook-Pro-de-riyad tests-mocha % npx mocha exo3.spec.js


 ex. 2
   ✔ should work with empty array
   ✔ should apply function for all values
   ✔ should return the same array


 3 passing (10ms)
```

Dans cet exercice, nous prenons une fonction de “mapping” de données et un tableau. Cette fonction de mapping va venir s’appliquer sur l’ensemble des éléments de notre collection.

Exercice 4:

```
riyad@MacBook-Pro-de-riyad tests-mocha % npx mocha exo4.spec.js


 ex. 4 parseCsvImperative(csvText)
   ✔ should ignore the header line (38ms)
   ✔ should have [username, realName, website, projectName] on each contribution (52ms)
   ✔ should parse username, realName and projectName (38ms)
   ✔ should set website to null if not provided
   ✔ should correctly parse current user data

 ex. 4 parseCsvFunctional(csvText)
   ✔ should ignore the header line
   ✔ should have [username, realName, website, projectName] on each contribution (38ms)
   ✔ should parse username, realName and projectName (39ms)
   ✔ should set website to null if not provided (50ms)
   ✔ should produce the same result as parseCsvImperative (216ms)


 10 passing (670ms)
```

Dans ces exercices 4 et 5, nous avons d'abord abordé la récupération et le traitement de données au format CSV. L'objectif était de transformer chaque ligne de ce CSV en un objet JavaScript avec des propriétés explicites : username, realName, website, et projectName. Deux versions de la fonction de parsing ont été créées : une approche impérative, avec des boucles et des variables temporaires, et une approche fonctionnelle, privilégiant l'usage de méthodes comme map et filter.

Exercice 5 :

Pour ce test nous avons décider de "mocker" la source de données, nous reprenons certaines des données du csv cependant nous interrogeons pas directement le csv

```
riyad@MacBook-Pro-de-riyad tests-mocha % npx mocha exo5.spec.js


  ex. 5 pullAndAnalyzeCsv()
    ✔ should compute various statistics about contributors


  1 passing (17ms)
```

L'exercice 5 consisté a analysé des données pour compter les contributeurs uniques, calculer la moyenne des longueurs de leurs noms, identifier les projets les plus contribué, et dégager des statistiques comme le contributeur le plus actif ou le TOP 10 des projets.