# Exercice noté 04

## Un exemple de decomposition fonctionelle

Vous devez concevoir et coder une fonction nommée "romain" qui fait la conversion d'un nombre entier entre 1 et 3999 en un texte correspondant à la numération romaine de ce nombre.

La numérotation romaine d'un nombre `n` se décompose en une partie qui représente (dans la numérotation arabe décimale) le chiffre des milliers, suivie d'une partie qui représente le chiffre des centaines, suivie d'une partie qui représente le chiffre des dizaines, et finalement une partie qui représente le chiffre des unités.

Ainsi la conversion du nombre 439, qui a 0, 4, 3, et 9 comme chiffres arabes des milliers, centaines, dizaines et unités, aura 4 parties : "", "CD", "XXX", "IX" qui seront concaténées. Chaque partie contient uniquement :

- les lettres I, V et X pour les unités (par exemple "VIII" pour 8)
- les lettres X, L et C pour les dizaines (par exemple "XL" pour 4)
- les lettres C, D et M pour les centaines (par exemple "CM" pour 9)
- la lettre M pour les milliers (par exemple "MM" pour 2)
- Si plusieurs lettres semblables sont écrites les unes derrière les autres, le nombre qui la représente s'ajoute (autant de fois qu'il y a de lettres).
- Toute lettre d'unité placée immédiatement à la gauche d'une lettre plus forte qu'elle, indique que le nombre qui lui correspond doit être retranché au nombre qui suit (ex. IV = 4, XC = 90, CD = 400).
- Les lettres I, X, C ne peut pas être employée quatre fois consécutivement sauf M.
- Les symboles V, L et D ne se répètent jamais.
- La soustraction se fait en mettant un I à gauche d'un V ou d'un X, en mettant un X à gauche d'un L ou d'un C ou un C à gauche d'un D ou d'un M.

Par exemple l'appel `romain(439)` doit retourner le texte "CDXXXIX" et `romain(2948)` doit retourner le texte "MMCMXLVIII". Vous devez faire une décomposition en sous-fonctions pour que votre code soit facile à comprendre et qu'il évite la duplication de code similaire. Vous devez avoir 4 fonctions au total dans votre code :

- `repeter`
- `chiffre`
- `romain`
- `testRomain`

La fonction `repeter` prend deux paramètres, un entier `n` et un texte `t`, et retourne un texte composé de `n` répétitions du texte `t` (par exemple, l'appel `repeter(3,"I")` doit retourner "`III`").

La fonction `chiffre` s'occupe d'encoder en numérotation romaine un seul chiffre du nombre décimal. Cette fonction traite 4 cas :

- chiffre de 0 à 3
- chiffre 4
- chiffre de 5 à 8
- chiffre 9

Pour les cas qui couvrent plus d'un chiffre (0 à 3 et 5 à 8), utilisez la fonction `repeter` pour répéter un certain nombre de fois l'unité (I, X, C ou M dépendant de la position du chiffre du nombre

décimal). Par exemple, `chiffre (439, 100, "C", "D", "M")` doit retourner le texte "CD" car le chiffre des centaines, c'est à-dire 4, est 5 -1 et le chiffre romain pour 500 est "D" et le chiffre romain pour 100 est "C". Vos tests unitaires doivent tester au moins une fois la conversion de chaque chiffre entre 0 et 9.

La fonction `testRomain` doit faire des tests unitaires bien choisis des fonctions `repeter`, `chiffre` et `romain`.

En plus de faire correctement la conversion demandée, votre code doit être correctement indenté, il doit contenir des commentaires explicatifs et des identificateurs significatifs. Il est à noter que votre code ne fait que définir des fonctions et exécuter les tests unitaires. Votre code ne doit pas faire d'entrée-sortie (pas d'appel à `print`, `alert` ou `prompt`). Il faut s'imaginer que votre code sera utilisé par un autre programmeur dans un logiciel plus gros (par exemple un logiciel de traitement de texte qui doit numéroter les pages en numérotation romaine).

Dans codeBoot, sauvez votre code sous le nom de fichier "romain.js" (il faut double cliquer sur le nom par défaut, puis entrer le nouveau nom, puis taper la touche RETURN).