**Sujet de l’activité**

Dans cette partie de l'activité, vous devez écrire des fonctions de manipulation de codes génétiques en utilisant la structure de données que vous souhaitez. Il n'y pas de contrainte particulière dans l'exercice et vous pouvez utiliser librement la bibliothèque standard. Cet exercice n'est pas un exercice de biologie et nous utiliserons une représentation et des opérations qui ne font qu'approximer ce qui se passe réellement.

Un code génétique est une séquence de codons. Chaque codon correspond à un acide aminé parmi un ensemble de quatre possibles généralement représentés par les lettres A, C, G, et T. Un code génétique est donc une séquence dont tous les éléments sont l'une de ces quatre lettres.

Vous devez écrire des fonctions qui créent de nouveaux codes génétiques pour toute opération. Tous les codes créés sont donc immuables et gardent leur valeur après une opération les utilisant comme opérande. La seule exception est la fonction code\_ajoute, qui modifie le code génétique passé en paramètre en lui ajoutant un codon.

## **1. Position et longueur**

Dans vos codes génétiques, comme pour les tableaux en C, les caractères successifs seront numérotés de 0 à la taille du code moins un.

Dans toutes les fonctions que vous avez à écrire, pour un code donné c de longueur taille\_code(c), dont les codons sont numérotés de 0 à taille\_code(c)-1, le paramètre position est un entier relatif quelconque qui désigne :

* la position qui précède le caractère de numéro position si position appartient à [0;taille\_code(c)-1] ;
* la fin de c si position est supérieur ou égal à taille\_code(c) ;
* la position 0 si position est négatif.

## **2. Fonctions à écrire**

Vous devez écrire les fonctions suivantes :

* **alloue\_code** : crée un nouveau code génétique vide ;
* **libere\_code** : libère un code génétique donné, précédemment alloué par une autre fonction de manipulation des codes génétiques ;
* **taille\_code** : renvoie la longueur (nombre de codons) du code génétique passé en paramètre ;
* **code\_element** : renvoie la valeur du codon ('A', 'C', 'G' ou 'T') de position donnée. Si la position donnée est égale à la fin du code, la valeur à renvoyer est ' ' ;
* **code\_ajoute** : ajoute le codon donné ('A', 'C', 'G' ou 'T') à la fin du code génétique donné. Si le codon donné n'est pas 'A', 'C', 'G', ou 'T', ou si la taille maximale d'un code a été atteinte, ne fait rien ;
* **sous\_code** : renvoie la plus petite position à laquelle il est possible de trouver une occurence du code donnée en second argument (b) dans le code donné en premier argument (a). Renvoie -1 si a ne contient pas b ;
* **coupe\_code** : crée deux nouveaux codes placés dans les deux dernier arguments (b et c), dont la valeur respective correspond aux morceaux du code donné en premier argument (a) coupé en deux à la position donnée (position). Autrement dit, b est un nouveau code contenant les codons de a dont la position est dans l'intervalle [0;position[ et c est un nouveau code contenant les codons de a dont la position est dans l'intervalle [position; taille\_code(a)[ ;
* **combine\_codes** : renvoie un nouveau code dont les codons correspondent à la concaténation des codons du premier argument (a) et du second argument (b), dans cet ordre. Si le code résultant dépasse la taille maximale, il est tronqué de manière à ne plus dépasser.

## **3. Utilisation de la mémoire**

Dans vos fonctions, et pour cette partie uniquement, vous pouvez supposer que la taille maximale d'un code génétique est 1024. En cas d'erreur d'allocation, alloue\_code devra renvoyer NULL. NULL devra être interprété par toutes vos fonctions comme un code vide.