**Sujet de l’activité**

Dans cette partie de l'activité, vous devez écrire les mêmes fonctions que dans l'activité précédente mais sans créer de nouveaux codes dans les opérations. À la place, vous devrez utiliser un des codes passés en paramètre en le redimensionnant au besoin.

Plus précisément, cela concerne les fonctions code\_ajoute (qui était déjà ainsi dans la première partie), coupe\_code (qui doit maintenant modifier a et le placer dans b, tandis que c reste un nouveau code à créer) et combine\_codes (qui doit maintenant modifier a et le renvoyer). Ces modifications d'un code passé en paramètre devront se faire en le redimensionnant au besoin.

En outre, dans cette partie, il y a deux différences concernant la mémoire :

* Il n'y a plus de limite de taille : vos codes devront grossir sans limite, du moment que les allocations mémoire réussissent. En cas d'erreur d'allocation, le comportement des fonctions devra être similaire à ce qui était demandé dans la partie précédente : alloue\_code devra renvoyer NULL, tandis que code\_ajoute, coupe\_code et combine\_codes devront laisser inchangés les codes qui leur sont passés en paramètre.
* Il faudra maintenant stocker les codons de manière compacte : comme il y a 4 codons différents, 2 bits suffisent à stocker un codon. Il vous faudra donc stocker 4 codons par octet que vous allouerez.

Attention, si vous voulez valider tous les tests, il faudra redimensionner vos codes de manière amortie : typiquement en doublant la taille allouée chaque fois qu'une réallocation est nécessaire et sans jamais avoir une quantité de mémoire allouée dépassant le double du nombre de caractères à stocker.