# 1 Première partie, `ManipSeqSimples.c`

## 1.1 Fonctions de manipulation des séquences

Vous écrirez les fonctions suivantes dans le fichier **`ManipSeqSimples.c`**. Leurs prototypes vous sont donnés dans le `ManipSeqSimples.h`.

```
char *NewSeq(int lg);
char *ReallocSeq(char *seq, int lg);
void FreeSeq(char *seq);


void AfficheSeq(char seq[], int lg);
void InitSeqAlea(char seq[], int lg);
float All_GC(char seq[], int lg, float *GC1, float *GC2, float *GC3);

int estStart(char *seq);
int estStop(char *seq);
char Nt_Complementaire(char nt);


char *BrinComplémentaire(char *seq, int lg);
```

- La fonction `NewSeq` alloue un tableau de `lg` caractère et le retourne.
- La fonction `ReallocSeq` réalloue le tableau de caractères `seq` pour qu'il puisse contenir `lg` caractères, et le retourne.
- La fonction `FreeSeq` libère le tableau de caractères `seq`.
- La fonction `InitSeqAlea` remplie la chaine de caractères `seq` de `lg` nucléotides tirés aléatoirement.
- La fonction `AfficheSeq` affiche la chaine de caractères `seq` de longueur `lg`.
- La fonction `All_GC` calcule et retourne la proportion de `G` et `C` dans la séquence `seq` de longueur `lg` mais aussi la proportion de GC en 1ere, 2e et 3e position grace aux arguments GC1, GC2 et GC3.
- La fonction `estStart` retourne 1 si le codon est un codon *start* et 0 sinon. La fonction `estStop` retourne 1 si le codon est un codon *stop* et 0 sinon.
- La fonction Nt_Complementaire retourne le nucléotide complémentaire du nucléotide passé en argument.
- La fonction `BrinComplémentaire` alloue un tableau de `lg` caractère, le rempli avec le complémentaire de *seq* et le retourne.

Pour tester vos fonctions, vous générez une séquence aléatoire de 1000 nucléotides, puis sont complémentaire. Vous calculerez et afficherez leurs taux de GC et compterez et afficherez le nombre de codons *start* et *stop*. Vous n'oublierez pas de libérer toute la mémoire allouée.

## 1.2 Une séquence est-elle biaisée ? (facultatif)

Les génomes sont souvent biaisés dans leur composition en ATGC. Cela est en partie dû à la fréquence d'utilisation des différents acides aminés dans les protéines, mais pas seulement. Ainsi, on

peut remarquer que le biais est plus fort en 3e position des codons des zones codantes. Pour prendre en compte cette particularité nous vous proposons de calculer un $\chi^2$ pour chaque ORF entre le nombre observé de GC à chaque position du codon et le nombre attendu selon la composition globale en GC du génome. Vous écrirez une fonction `float calcChi2Conformite(char *seq, int lg, float GCGlobal )` calculant ce $\chi^2$. Vous modifierez votre `main` pour tester vos fonctions.