Fiche de TP N°3

I. Alignement par les matrices de similarité :

1. Écrire un programme qui recherche sur NCBI par identifiant les séquences suivantes et les retourne sous forme de chaînes de caractères :
   - NM_001101.5 (Homo sapiens actin beta)
   - AC035147.4 (Homo sapiens chromosome 5 clone CTD-2309M13)
2. Visualiser la matrice BLOSUM62 :

```
from Bio.Align import substitution_matrices

blosum62 = substitution_matrices.load("BLOSUM62")
```

3. Calculer le score d'alignement des séquences précédentes en utilisant cette matrice (GAP=0), puis refaire le calcul en utilisant la matrice de similarité ci-dessous :

	–	A	C	G	T
–	0	-3	-3	-3	-3
A	-3	2	-1	-1	0
C	-3	-1	2	0	-1
G	-3	-1	0	2	-1
T	-3	0	-1	-1	2

II. Programmation dynamique :

1. Créer les deux fonctions :

- alignement_global(seq1, seq2, score_table) qui retourne la table de programmation dynamique globale et une table de score
- alignement_global_resultat(seq1, seq2, score_table) qui affiche le score et un alignement qui lui correspondant

2. Tester sur les deux séquences utiliser dans la partie 1 avec :

Sub (Substitution) = −1 Gap (Pénalité de gap) = −2

Id (Identité) = +1 Ext (Pénalité d'extension de gap) = −1

4. Utiliser la fonction globalms pour effectuer un alignement global et comparer le résultat avec celui obtenu à l'aide des fonctions précédentes

```
from Bio import pairwise2

alignments = pairwise2.align.globalms(sequence1, sequence2, match, mismatch, gap_open, gap_extend)
```

4. Refaire les étapes pour l'alignement locale et tester sur les mêmes séquences et paramètres :

   - alignement_local(seq1, seq2, score_table) qui retourne la table de programmation dynamique locale avec une table de score
   - alignement_local_resultat(seq1, seq2, score_table) qui affiche le score du meilleur alignement local et un alignement lui correspondant

Fonction Biopython pour l'alignement local :

```
alignments = pairwise2.align.localms(sequence1, sequence2, match, mismatch, gap_open, gap_extend)
```