Exercice 1 : Scripts perl

1. Ecrire un script perl permettant de calculer et d'afficher la moyenne de deux entiers passés en paramètres.
2. Ecrire un script perl prenant en entrée un nom et un prénom et retournant à l'écran la chaîne "Bonjour prénom nom".
3. Modifier votre script de façon à stocker le résultat dans un fichier nommé "a.out".
4. Ecrire un script perl permettant de trier une liste d'entiers dont la valeur est comprise entre 1 et n, n étant l'argument du script. Pour créer cette liste d'entiers, vous pourrez utiliser la fonction `rand`, par exemple `int(rand(10))` retourne un entier compris entre 0 et 9.

Exercice 2 : Expressions régulières

1. Ecrire un programme perl demandant à l'utilisateur d'entrer un nombre (réel ou entiers, positif ou négatif) et affiche le type de nombre saisi.
2. Ecrire un programme perl prenant en entrée une suite de caractère et vérifie :
   (a) s'il s'agit d'une plaque d'immatriculation française,
   (b) (si oui) s'il s'agit d'une plaque d'un département lorrain,
   (c) (si oui) affiche la plaque complétée par le nom de ce département en toutes lettres.
3. Ecrire un programme perl permettant de compter le nombre de mots, lignes et caractères d'un fichier texte passé en paramètre (analogie avec la commande `wc`).

Exercice 3 : Manipulation de références

Soient les variables `a` et `b` définies comme suit :

```
@a = ([1,2,3],[4,5,6],[7,8,9]);
$b = [[1,2,3],[4,5,6],[7,8,9]];
```

1. Que contiennent les variables `a` et `b` ?
2. Comment remplacer la valeur 8 dans le tableau `a` par la chaîne "huit" ?
3. Idem dans le tableau associé à `b`.

Exercice 4 : Tableaux associatifs et références

On suppose qu'on dispose d'un fichier texte dont le contenu a la forme suivante :

```
Chicago, USA
Paris, France
Brussels, Belgium
Nancy, France
...
```

1. Ecrire un programme perl qui parcourt ce fichier et affiche en sortie la liste des pays par ordre alphabétique, et pour chaque pays, la liste des villes qui y sont situées :

```
Belgium: Brussels, Charleroi, Liège
France: Nancy, Paris
...
```

2. Modifier votre programme afin d'utiliser des fonctions pour le tri et l'affichage (si ce n'est déjà fait).

Exerice 5 : Modules

Ecrire un module perl permettant de faire les traitements suivants sur un fichier texte :
- remplacement des majuscules par des minuscules,
- retour à la ligne après chaque "."

Exercice 6 : Tableaux associatifs et références

1. Ecrire un programme prenant en entrée (a) un fichier contenant une liste de mot et (b) un fichier contenant une liste de suffixes. Ce programme triera les mots par suffixe, avant de les afficher à l'écran.
2. Etendre le programme précédent afin de gérer également (et de stocker), pour chaque suffixe, le nombre de mots trouvés dans le texte.

Exercice 7 : Traitement de séquences

Téléchargez le fichier genbank et fasta de la séquence ayant un numéro d'accès suivant : `NM_207040` (gènes codant pour un facteur de transcription chez l'homme).

1. Ecrire un programme qui affiche le complément de la séquence.
2. Ajouter à ce programme, l'affichage de la longueur de la séquence.
3. Lister tous les numéros de référence PUBMED, le nom du premier auteur associé et la date d'édition.
4. Extraire la nature du génome où la séquence de ce gène a été retrouvée (génome linéaire, circulaire).
5. Afficher les 30 premiers acides aminés de la séquence avec un code à trois lettres :

```
A : Ala     H : His     P : Pro     W : Trp
C : Cys     I : Ile     Q : Gln     Y : Tyr
D : Asp     K : Lys     R : Arg
E : Glu     L : Leu     S : Ser
F : Phe     M : Met     T : Thr
G : Gly     N : Asn     V : Val
```

Exercice 8 : BioPerl

1. Ecrire un programme BioPerl prenant en entrée un fichier au format genbank (par exemple le fichier `NM_207040` précédemment téléchargé), et en extrait les informations suivantes (utiliser `Bio::Perl`) :
   (a) identifiant de séquence
   (b) longueur de séquence
   (c) 10 premières bases de la séquence
2. Ecrire un programme BioPerl qui demande à l'utilisateur un identifiant de séquence, la télécharge au format genbank, et en extrait les informations suivantes (utiliser `Bio::Seq` et `Bio::SeqIO`) :
   (a) séquence
   (b) longueur de la séquence
   (c) complément de la séquence
   (d) CDS
3. Etendez le programme précédent de manière à produire en sortie un fichier au format fasta contenant la séquence téléchargée (utiliser `Bio::SeqIO`).
4. Ecrire un programme BioPerl permettant d'exécuter une requête sur une base de donnée GenBank (utiliser `Bio::DB::Query::GenBank`).