DM n°1 - Pointeurs, références, tableaux et chaînes de caractères.

Les codes `NOM_tri.ml`, `NOM_voyelles.ml`, `NOM_motif.c`, `NOM_motif.ml` et `NOM_matrices.c` sont à rendre sur Moodle.
Les éléments écrits sur papier sont à rendre en prenant une photographie ou un scan que vous déposerez sur Moodle également.

Exercice 1 (Tri à bulles).
Dans un script OCaml nommé `NOM_tri.ml`, écrire une fonction `tri_a_bulles` qui trie un tableau par ordre croissant de ses valeurs.
Vous laisserez des traces de vos tests (au moins 3) dans le script.

Exercice 2 (Voyelles et OCaml).
Dans cet exercice, on ne traitera que le cas de textes en minuscules sans aucun accent. Les deux fonctions qui suivent seront codées dans un script OCaml nommé `NOM_voyelles.ml`.

1. Écrire une fonction `est_voyelle` qui prend en entrée un caractère et retourne vrai si celui-ci est une voyelle, et faux sinon.
   Vous laisserez des traces d'au moins 3 tests dans le script.
2. Écrire une fonction `compte_voyelles` qui retourne le nombre de voyelles présentes dans une chaîne de caractères.
   Vous laisserez des traces d'au moins 3 tests dans le script.

Exercice 3 (Repérage d'un motif dans un texte).

1. Sur papier, écrire le diagramme entrée/sortie puis le pseudo-code d'un algorithme de repérage d'un motif (suite de caractères) dans un texte. Cet algorithme donne le nombre d'occurrences [a] du motif dans le texte et retourne un pointeur sur sa première occurrence. Voici un exemple d'exécution d'un tel algorithme :

```
golivier@ordiprof:~/doc/MPII/DM1$ gcc motif.c -o motif -Wall
golivier@ordiprof:~/doc/MPII/DM1$ ./motif "ma maman m'amadoue" "ama"
Le motif ama a été repéré 2 fois dans le texte:
"ma maman m'amadoue"
Première occurence: aman m'amadoue
golivier@ordiprof:~/doc/MPII/DM1$ ./motif "l'arrivee des riverains pres de la riviere" "riv"
Le motif riv a été repéré 3 fois dans le texte:
"l'arrivee des riverains pres de la riviere"
Première occurence: rivee des riverains pres de la riviere
golivier@ordiprof:~/doc/MPII/DM1$
```

2. Sur papier, préparez au moins 5 tests permettant de tester votre code dans une grande variété de cas.
3. Écrire en C un programme `NOM_motif.c` qui repère un motif (série de caractères) dans un texte. Le texte et le motif ciblé seront fournis en ligne de commande par l'utilisateur. Le code comprendra une fonction indépendante `repere_motif` qui prendra en entrée le motif cible et le texte dans lequel rechercher ce motif, et retournera le nombre d'occurrences du motif, ainsi qu'un pointeur localisant la première occurrence du motif.
   Si le texte ou le motif contiennent des caractères blancs, l'utilisateur devra donner son texte sur la ligne de commande en prenant soin de l'entourer de double guillemets comme dans l'exemple ci-dessus.
   Vous êtes autorisé à utiliser la fonction `strlen` de la bibliothèque `string.h` si vous le souhaitez.
   Pensez à appliquer la méthode des petits pas, à rendre votre code clair, aéré et lisible.
4. Testez votre programme sur les tests préparés sur papier. Une fois la phase de test validée, veillez à appliquer une programmation défensive en faisant en sorte que votre code soit résilient aux erreurs de l'utilisateur ou aux cas limites. Vous pouvez appliquer les principes de la programmation défensive en utilisant la bibliothèque `assert`.
5. Quelle est la complexité de ce code en nombre d'opérations arithmétiques et tests (tout confondus), en fonction du nombre de caractères N du texte et M du motif ?
6. Dans un script OCaml nommé `NOM_motif.ml`, écrivez une fonction `repere_motif` qui retourne le nombre d'occurrences d'un motif dans un texte.

L'algorithmique des textes est presque une discipline à part entière. Nous verrons au deuxième semestre des algorithmes très astucieux et beaucoup moins coûteux pour repérer des motifs dans un texte !

a. occurrences = apparitions

Exercice 4 (Linéarisation d'un tableau multi-dimensionnel).
En mathématiques, une matrice est un tableau de valeurs numériques à deux dimensions. La taille d'une matrice est un couple d'entiers naturels strictement positifs (N, M) correspondant respectivement au nombre de lignes et au nombre de colonnes de la matrice. On dit alors que la matrice est de taille $N \times M$. Un élément de la matrice est repéré par son double indice (i, j) où $1 \leq i \leq N$ et $1 \leq j \leq M$.

Écrire un programme `NOM_matrices.c` qui

- prend en entrée deux valeurs entières strictement positives N et M correspondant respectivement au nombre de lignes et au nombre de colonnes de la matrice ;
- alloue l'espace mémoire nécessaire au stockage d'une matrice $N \times M$ à valeurs entières ;
- calcule la matrice dont la valeur m_{ij} associée à l'indice (i, j) est donnée par la formule suivante :

$$m_{ij} = \begin{cases} (i-1)+(j-1) & \text{si } j \geq i \\ 0 & \text{sinon} \end{cases}$$

  Indication : Attention, au niveau informatique, l'indexation des tableaux commence à 0 !
- affiche cette matrice à l'écran sous la forme d'un tableau bidimensionnel.

Voici un exemple d'exécution du programme :

```
golivier@ordiprof:~/doc/MPII/TP5$ gcc matrices.c -o matrices -Wall
golivier@ordiprof:~/doc/MPII/TP5$ ./matrices 2 3
Matrice calculée de taille 2 par 3:
  0   1   2
  0   2   3

golivier@ordiprof:~/doc/MPII/TP5$ ./matrices 6 6
Matrice calculée de taille 6 par 6:
  0   1   2   3   4   5
  0   2   3   4   5   6
  0   0   4   5   6   7
  0   0   0   6   7   8
  0   0   0   0   8   9
  0   0   0   0   0  10

golivier@ordiprof:~/doc/MPII/TP5$ ./matrices 10 7
Matrice calculée de taille 10 par 7:
  0   1   2   3   4   5   6
  0   2   3   4   5   6   7
  0   0   4   5   6   7   8
  0   0   0   6   7   8   9
  0   0   0   0   8   9  10
  0   0   0   0   0  10  11
  0   0   0   0   0   0  12
  0   0   0   0   0   0   0
  0   0   0   0   0   0   0
  0   0   0   0   0   0   0
```

- Structurez votre programme en le découpant en trois fonctions : `allouer_matrice`, `remplir_matrice` et `afficher_matrice`
- Stockez votre matrice dans un grand tableau uni-dimensionnel (cf cours séquence 4)
- La fonction d'affichage de la matrice doit être appelée dans le main (pour vous obliger à faire en sorte que la matrice survive à la fonction `allouer_matrice`)
- Testez votre code, par exemple sur les mêmes exemples que moi
- Pensez à bien regarder les corrigés du TP5 pour essayer de "coller" aux attendus et aux bonnes pratiques de programmation.
- Pour l'affichage, on pourra utiliser le format `%3d` au lieu de `%d` pour forcer l'affichage de la valeur sur 3 emplacements quelque soit le nombre de chiffres du nombre, et ainsi garantir un bel alignement des valeurs de votre tableau.