Programmation C, TP 2

Exercice 1

Ecrire un programme qui détermine tous les diviseurs plus grands que 1 d'un nombre entier saisi.

Exercice 2

Un nombre entier est parfait s'il est égal à la somme de ses diviseurs (sauf lui-même).
Ex : $6 = 1 + 2 + 3$ est parfait.

- Ecrire une fonction `somme_div` qui retourne la somme des diviseurs d'un nombre passé en paramètre.
- Ecrire une fonction `parfait` qui teste si un nombre passé en paramètre est parfait et qui retourne 1 s'il l'est et 0 sinon.
- Ecrire un programme principal qui affiche tous les nombres parfaits inférieurs à une certaine limite.

Exercice 3

- Ecrire une fonction `void init(int tableau[], int taille)` qui remplit le tableau donné en paramètre avec des entiers aléatoires (en utilisant la fonction `rand()`).

- Ecrire une fonction `void tri(int tableau[], int taille)` qui trie un tableau d'entiers.

Exercice 4

On propose dans cet exercice d'implementer une calculatrice rudimentaire. Le programme execute une boucle avec les opérations suivantes :

1. Lire une ligne saisie par l'utilisateur.
2. Repérer l'opérateur (les opérateurs autorisés sont +,-,*,/).
3. Repérer les opérandes (les opérandes sont des entiers). On considère que les chiffres à gauche de l'opérateur forment le premier opérande, et les chiffres à droite de l'opérateur forment le second opérande.
4. Calcul du résultat.
5. Affichage du résultat à l'écran.

Par ailleurs, la calculatrice doit vérifier si la ligne saisie par l'utilisateur est au bon format : `operande1 opérateur operande2`. Si l'opérateur est manquant ou que l'un des opérandes n'est pas composé de chiffres, la calculatrice doit retourner une erreur. Par exemple :

- la ligne `1324fdg3+43` est incorrecte car le premier opérande `1324fdg3` n'est pas un entier.
- la ligne `13%24` est incorrecte car `%` n'est pas un opérateur valide.

Exercice 5

Cet exercice propose de manipuler des chaines de caractères. On utilisera le type `typedef char mot[TAILLE_MAX]` avec `TAILLE_MAX` une constante entière préalablement définie.

Pour cet exercice, vous n'utiliserez aucune des fonctions `strcpy, strcmp, strncmp`. Vous pourrez néamoins utiliser `strlen` (définie dans `string.h`).

Questions :

- Ecrire et tester une fonction `void miroir(mot dest, mot src)` rangeant dans le mot `dest` le miroir (mot à l'envers) du mot `src`.

- Ecrire et tester une fonction `void majuscules(mot dest, mot src)` convertissant les lettres minuscules du mot `src` en majuscules et en sauvegardant le résultat dans le mot `dest` (les éventuels caractères autres que les minuscules ne sont pas modifiés.

- Ecrire et tester une fonction `int est_facteur_gauche(mot facteur_g, mot src)` qui vérifie si `facteur_g` est un facteur gauche de `src`.
  Par exemple, "poly" est facteur gauche de "polycopié".

- Ecrire et tester une fonction `int est_facteur(mot facteur, mot src)` qui vérifie si le mot `facteur` est un facteur de `src`.
  Par exemple, "poly", "copi", "polycop" sont des facteurs de "polycopié".

Exercice 6

On souhaite concevoir une librarie pour la manipulation de matrices d'entiers. Une matrice est un tableau d'entiers à deux dimensions. On propose donc d'utiliser la structure suivante pour representer la matrice :

```
typedef struct
     int **valeurs;
     int longueur;
     int largeur;
matrice;
```

Questions :

- Ecrire la fonction `matrice creer_matrice()` qui demande à l'utilisateur de saisir deux entiers corresondant aux dimensions de la matrice, puis alloue le tableau à deux dimensions (en utilisant la fonction `malloc`).
- Ecrire la fonction `matrice alea_matrice()` qui retourne une matrice remplie avec des entiers aléatoires compris entre 0 et 100 (à l'aide de la fonction `rand()`).
- Ecrire la fonction `void affiche_matrice(matrice m)` qui affiche dans la console la matrice donnée en paramètre.
- Ecrire la fonction `matrice addition_matrice(matrice mat1, matrice mat2` qui effectue l'addition de deux matrices : MAT1 = MAT1 + MAT2
- Ecrire la fonction `matrice entier_matrice(int x, matrice mat)` qui effectue la multiplication de la matrice MAT1 par un entier X : MAT1 = X * MAT1
- Ecrire la fonction `matrice multiplication_matrice(matrice mat1, matrice mat2)` qui effectue le produit de deux matrices : MAT1 = MAT1 * MAT2