Exercice 1 : Expérimentations avec les tableaux

Q. 1 Déclarer un tableau `tab` de taille 4, dans les cases 0, 1 et 2 du tableau écrire la valeur 0, imprimer le contenu de la 3-ième et 4-ième case.

Q. 2 En une seule ligne de code déclarer un tableau `tab2` de taille 4, y écrire les valeurs des entiers de 1 à 4.

Q. 3 Déclarer un 3-ième tableau de taille 4 `tab3`, recopier le contenu de `tab2` dans `tab3`.

Q. 4 Imprimer le contenu du tableau `tab3`.

Q. 5 Écrire une fonction permettant l'affichage du contenu du tableau `tab3`. Quelle est la signature de votre fonction ?

Q. 6 Déclarer un 4-ième tableau `tab4` de taille 5. Pouvez vous afficher `tab4` en utilisant la fonction de la question précédente ?

Dans le cas des arguments de fonctions, afin de généraliser et de ne pas avoir une fonction par taille de tableaux statiques, il est possible de remplacer dans les arguments de fonctions le type `int tab[3]` par `int tab[]` ne précisant donc pas la longueur du tableau statique. **ATTENTION**, ce ne peut être fait que dans les arguments d'une fonction, mais pas dans les déclarations de variables locales.

Q. 7 Définir une nouvelle fonction d'affichage du contenu d'un tableau.

Q. 8 Définir une fonction `void fois_deux_entier(int x)` multipliant le contenu de la variable `x` par 2 et stockant le résultat dans `x`. Définir une fonction `void fois_deux_tab(int t[1])` multipliant le contenu de la première case du tableau `t[0]` par 2 et stockant le résultat dans `t[0]`. Dans votre fonction `main` déclarer une variable `x` initialisée à 1, et un tableau `int t[1]` dont la première case est initialisée à 1. Comparer les valeurs de `x` et `tab[0]` après appels aux fonctions `fois_deux_entier` et `fois_deux_tab` sur `x` et `t`.

Q. 9 Définir une fonction permettant le calcul du tableau contenant le double de toutes les valeurs d'un tableau `tab` passé en argument à la fonction. On ne modifiera pas le tableau `tab`.

Exercice 2 : Itérations sur des tableaux en dimension 1

Q. 1 Donner une fonction `init_alea` prenant en argument un tableau et deux entiers `min` et `max` et l'initialisant avec des valeurs aléatoires choisies uniformément dans l'intervalle $[\![\texttt{min}, \texttt{max}]\!]$.

Q. 2 Donner une fonction `init_linspace` prenant en argument un tableau et deux flottants x et y et initialisant le tableau avec la suite des valeurs $x + i\frac{(y-x)}{n}$ où n est la longueur du tableau en argument.

Q. 3 Écrire une fonction `somme_tab` qui prend en paramètre un tableau d'entiers et calcule la somme des éléments contenus dans le tableau.

Q. 4 En déduire une fonction `moyenne_tab` qui calcule la moyenne des valeurs situées dans un tableau d'entiers passé en argument.

Q. 5 Écrire une fonction `minmax_tab` qui affiche la valeur minimale et la valeur maximale contenues dans un tableau de flottants.

Q. 6 Écrire une fonction `copie_tab` prenant en argument deux tableaux de flottants et recopiant le contenu du premier dans le deuxième.

Q. 7 Écrire une fonction `injection` prenant en argument un tableau d'entiers et vérifiant qu'aucune valeur n'apparaît deux fois dans le tableau.

Q. 8 Écrire une fonction `nb_inversion` comptant le nombre d'inversions dans un tableau d'entier, c'est à dire le nombre de paires (i, j) telles que $i < j$ et `t[`i`]` $>$ `t[`j`]`.

Exercice 3 : Tableaux et chaînes de caractères

En C, une chaîne de caractères est représentée par un tableau de caractères (`char[]`). Dans ce langage, un tableau ne connaît pas sa taille, (c'est pour cela que toutes les fonctions précédentes prenaient en argument la taille du tableau), mais pour ce qui est des chaînes de caractère, c'est un peu différent : elles terminent systématiquement par un caractère de fin de chaîne (invisible), noté '\0'. Ce dernier est donc situé dans le tableau juste après le dernier caractère.

Q. 1 Déclarer un tableau `s` de 10 caractères `char`. Le remplir avec une chaîne de 4 caractères au moyen d'un `scanf("\%s", &s);`, puis imprimer les valeurs numériques des 10 caractères de la chaîne au moyen par exemple d'un `printf("\%d", s[2];`. Vérifier empiriquement le paragraphe ci-dessus.

Q. 2 Définir une fonction `longueur_chaine` prenant en argument une chaîne de caractères (un tableau statique de caractères) et retournant sa longueur.

Q. 3 Définir une fonction `est_palindrome` prenant en argument une chaîne de caractères et renvoyant si oui ou non cette chaîne est un palindrome. ♣

Q. 4 Écrire une fonction `compte` qui compte le nombre d'occurrences d'un caractère cr dans une chaîne.

Q. 5 En déduire une fonction `affiche_occur` qui utilise la fonction compte pour afficher le nombre d'occurrences de chaque caractère d'une chaîne. On fera bien attention à ne pas afficher plusieurs fois le nombre d'occurences du même caractère d'un mot.

Q. 6 Écrire une fonction calculant le caractère le plus fréquent dans un texte.

Exercice 4 : Itérations sur des tableaux en dimension 2

On appelle matrice un tableau à deux dimensions. Cela correspond mathématiquement à une suite finie $(u_{i,j})_{0 \leqslant i < n \text{ et } 0 \leqslant j < m}$. On représentera graphiquement une telle suite de la manière suivante :

$u_{0,0}$	$u_{0,1}$	$\dots$	$u_{0,m-1}$
$u_{1,0}$	$u_{1,1}$	$\dots$	$u_{1,m-1}$
$\dots$	$\dots$	$\dots$	$\dots$
$u_{n-1,0}$	$u_{n-1,1}$	$\dots$	$u_{n-1,m-1}$

Il est possible de représenter une matrice en C par l'intermédiaire d'un tableau à deux dimensions (autrement dit : un tableau de tableaux) :

```c
int matrix[10][20]; // une matrice 10 * 20
```

♣. *Texte dont la succession des lettres ou des chiffres est la même quand on le parcourt de gauche à droite ou de droite à gauche.*

Contrairement aux tableaux, la taille de la matrice devra être précisée dans la signature des fonctions concernées. Par exemple, la fonction g reçoit une matrice m d'entiers (de taille 10*20) en paramètre :

```
1 void g(int m[10][20]);
```

Q. 1 Écrire une fonction qui affiche une matrice 3×3.

Q. 2 Vous disposez d'une matrice 3 × 3. Demandez à l'utilisateur de la remplir de chiffres entre 1 et 9 (chaque chiffre devra apparaître une et une seule fois), puis vérifiez si c'est un carré magique (les sommes des chiffres sur les lignes, les colonnes et les diagonales sont égales).

Q. 3 Écrire une fonction prenant en argument une matrice qui échange le triangle inférieur avec le triangle supérieur dans un tableau à deux dimensions. Par exemple :

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

devient

1	5	9	13
2	6	10	14
3	7	11	15
4	8	12	16

Q. 4 Écrire une fonction qui prend en paramètre une matrice 10 × 10 (initialisée avec des 0), et la remplit avec les valeurs d'un triangle de Pascal de 10 lignes. Les cases `m[i][j]` de la matrice, dénotée $m_{i,j}$, devront donc vérifier :

— $m_{i,0} = 0$;
— $m_{i,j} = m_{i-1,j-1} + m_{i-1,j}$;

Par exemple pour le cas d'une matrice 5×5 :

1	0	0	0	0
1	1	0	0	0
1	2	1	0	0
1	3	3	1	0
1	4	6	4	1

Exercice 5 : Crible d'Ératosthène

Dans cet exercice, nous nous intéressons à l'implantation de l'algorithme du crible d'Ératosthène permettant de trouver tous les nombres premiers dans un intervalle $[\![2, n]\!]$. On pourra regarder l'animation suivante Gif wikipedia décrivant parfaitement l'algorithme.
L'algorithme procède par élimination successive des nombres que l'on a découvert comme n'étant pas premiers. Initialement tous les nombres entiers de l'intervalle $[\![2, n]\!]$ sont marqués comme étant potentiellement premiers puis à chaque nombre certainement premier que l'on découvre on marque tous ses multiples comme n'étant pas premiers, on passe ensuite au nombre suivant marqué comme potentiellement premier et on recommence. Quand on atteint un nombre n marqué comme potentiellement premier, c'est qu'aucun nombre le divisant n'a été croisé dans les itérations précédentes, n est donc bien premier.
Afin de garder en mémoire si un entier est marqué comme potentiellement premier ou pas on utilisera un tableau de booléen. On utilisera la librairie `stdbool` fournissant les deux booléens `true`

et `false`. Un entier i sera donc marqué comme potentiellement premier si la case i du tableau contient `true` et comme certainement non premier si elle contient `false`.
Le tableau ci-dessous de taille 11 représente le fait qu'initialement chaque entier de l'intervalle $[\![0, 10]\!]$ est potentiellement premier. L'algorithme se déroule alors de la manière suivante.
{`true`, `true`, `true`, `true`, `true`, `true`, `true`, `true`, `true`, `true`, `true`**}** On marque initialement 0 et 1 comme n'étant pas premiers :
{`false`, `false`, `true`, `true`, `true`, `true`, `true`, `true`, `true`, `true`, `true`**}**
puis on itère le processus décrit ci-dessus.

- On arrive à l'entier 2 qui est potentiellement premier et donc premier, on marque ses multiples comme non premiers.
  {`false`, `false`, `true`, `true`, `false`, `true`, `false`, `true`, `false`, `true`, `false`**}**
- On arrive à l'entier 3 qui est potentiellement premier et donc premier, on marque ses multiples comme non premiers.
  {`false`, `false`, `true`, `true`, `false`, `true`, `false`, `true`, `false`, `false`, `false`**}**
- On arrive à l'entier 4 qui est non premier, le tableau est inchangé.
- On arrive à l'entier 5 qui est potentiellement premier et donc premier, on marque ses multiples comme non premiers.
  {`false`, `false`, `true`, `true`, `false`, `true`, `false`, `true`, `false`, `false`, `false`**}**
- On arrive à l'entier 6 qui est non premier, le tableau est inchangé.
- On arrive à l'entier 7 qui est potentiellement premier et donc premier, on marque ses multiples comme non premiers.
  {`false`, `false`, `true`, `true`, `false`, `true`, `false`, `true`, `false`, `false`, `false`**}**
- On arrive à l'entier 8 qui est non premier, le tableau est inchangé.
- On arrive à l'entier 9 qui est non premier, le tableau est inchangé.
- On arrive à l'entier 10 qui est non premier, le tableau est inchangé.

À la fin des itérations on remarque que le tableau indique la primalité de chaque nombre de l'intervalle $[\![0, 10]\!]$.

Q. 1 Implanter le crible d'Ératosthène.