



3. Que fait le code suivant ? On ne vous demande pas de paraphraser. Vous décrierez sa fonction et vous l'illustrerez par un exemple.

```

1 int mystere(char * c) {
2     char * oo = c ;
3     while ( *(c++) != '\0' ) ;
4     return c - oo - 1 ;
5 }

```

.....

.....

.....

.....

.....

.....

4. Écrire une fonction `echange`, qui prend en argument deux pointeurs vers des flottants et qui échange leurs valeurs. Par exemple, si `a` pointe vers la valeur 3.5 et `b` pointe vers la valeur 2.0, l'appel de `echange(a, b)` fera pointer `a` vers 2.0 et `b` vers 3.5.

.....

.....

.....

.....

.....

.....

5. Utiliser la fonction `echange` pour compléter le code suivant. À la fin, les valeurs de `x` et `y` doivent être permutées. Votre code ne doit pas dépasser une ligne.

```

1 float x = 2.0;
2 float y = 3.0;
3 /* ligne à compléter pour que x vaille 3.0 et y 2.0 */

```

.....

.....

## Problème (13 points)

On cherche dans cet exercice à implémenter une file à partir de deux tableaux. Ce n'est pas forcément optimal, mais c'est un prétexte pour faire du C. En pratique, la file est séparée en deux tableaux : un premier auquel on ajoute les éléments et un second, duquel on les retire. Lorsque le second tableau est vide, on transfère le contenu du premier vers celui-ci.

Exemple. Prenons la file abstraite suivante :  $\rightarrow \boxed{3} \boxed{5} \boxed{7} \boxed{2} \boxed{1} \rightarrow$  ; elle sera implémentée en C par les deux tableaux ci-dessous.

- entree : 

7	5	3							
---	---	---	--	--	--	--	--	--	--

 ← (notez comme l'ordre est inversée par rapport à la file)
- sortie : 

2	1								
---	---	--	--	--	--	--	--	--	--

 →

Les cases vides, représentent des valeurs indéfinies. Pour connaître à chaque instant le nombre de cases effectivement remplies on ajoute deux variables correspondant à des indices.

- indice\_entree represente la dernière case remplie du tableau entree (ici, la troisième case d'indice 2).
- indice\_sortie represente la première case vide du tableau sortie (ici aussi, la troisième case d'indice 2).

Ainsi, si j'ajoute l'élément 10 (à la fin de la file) et que je supprime l'élément du début de la file (ici, 1), j'obtiens comme nouvelle file  $\rightarrow \boxed{10} \boxed{3} \boxed{5} \boxed{7} \boxed{2} \rightarrow$  La file sera alors codée par :

- entree : 

7	5	3	10						
---	---	---	----	--	--	--	--	--	--

 ← (et indice\_entree vaut maintenant 3)
- sortie : 

2									
---	--	--	--	--	--	--	--	--	--

 → (et indice\_sortie vaut maintenant 1)

On vous donne la structure suivante définissant le type file.

```

1 typedef struct {
2     int entree[N] ;
3     int indice_entree ;
4     int sortie[N] ;
5     int indice_sortie ;
6 } file ;
    
```

1. Définir la valeur N à 10 à l'aide d'une macro (en utilisant le préprocesseur).

.....

.....

2. Que faut-il écrire comme code pour créer une nouvelle file sur la pile avec les indices correctement initialisés.

.....

.....

.....

.....

.....

3. Écrire une fonction `est_vide` qui prend une file en paramètre et renvoie un booléen (c'est-à-dire un entier) pour indiquer si la file est vide.

.....

.....

.....

.....

.....





