

*La notation tiendra compte de la lisibilité du code et de la clarté des explications.
 La barème n'est donné qu'à titre indicatif.
 Notes de cours, TD, TP autorisées. Autres documents interdits.*

Exercice 1 (5pts) Répondre par vrai ou faux aux questions suivantes. Barème: 0,5 pour une réponse correcte, -0,5 pour une réponse incorrecte et 0 en cas d'absence de réponse. Dans cet exercice, il n'est pas demandé de justification. Ici, Vrai signifie « toujours vrai » et Faux signifie « pas toujours vrai ».

- | | Vrai | Faux |
|---|--------------------------|---|
| 1. Les fichiers manipulés par les fonctions de la bibliothèque standard sont terminés par un caractère spécial EOF..... | <input type="checkbox"/> | <input type="checkbox"/> |
| 2. En l'absence d'initialisation explicite, une variable de type <code>int</code> est initialisée à 0.. | <input type="checkbox"/> | <input type="checkbox"/> |
| 3. On peut écrire des fonctions prenant un nombre variable de paramètres..... | <input type="checkbox"/> | <input type="checkbox"/> |
| 4. Si <code>p</code> désigne un pointeur, les expressions <code>(p+1)</code> et <code>(&p[1])</code> sont équivalentes..... | <input type="checkbox"/> | <input type="checkbox"/> |
| 5. Après la séquence: <code>char c[5];</code>
<code> c[3] = '\0';</code>
il est interdit d'écrire dans <code>c[4]</code> | <input type="checkbox"/> | <input type="checkbox"/> |
| 6. L'expression <code>(f() && g())</code> est équivalente à l'expression <code>(g() && f())</code> | <input type="checkbox"/> | <input type="checkbox"/> |
| 7. Pour créer un fichier objet (.o) à partir d'un source C (.c), il n'est pas nécessaire d'avoir défini toutes les fonctions avant leur utilisation dans le source..... | <input type="checkbox"/> | <input type="checkbox"/> |
| 8. L'expression <code>f(i++,i++)</code> s'évalue comme <code>f(1,2)</code> lorsque <code>i</code> vaut initialement 1.... | <input type="checkbox"/> | <input type="checkbox"/> |
| 9. La séquence suivante est incorrecte en C: <code>char c;</code>
<code> char p[10], *q;</code>
<code> p = q = &c;</code> | <input type="checkbox"/> | <input type="checkbox"/> |
| 10. Les deux séquences suivantes | | |
| Séquence 1 : <code>static int x;</code>
<code> x = 1;</code> | | Séquence 2 : <code>static int x = 1;</code> |
| sont équivalentes si elles sont placées au début d'une définition de fonction..... | | |

Exercice 2 (3pts) Écrire un programme qui lit un texte sur l'entrée standard et le réécrit sur la sortie standard, en remplaçant toutes les lettres majuscules par les minuscules correspondantes. Par exemple, la chaîne `Texte!` sera remplacée par la chaîne `texte!`. On pourra utiliser les fonctions `int isupper (int c)`; qui teste si `c` est une majuscule, `int islower (int c)`; qui teste si `c` est une minuscule, et `int tolower (int c)`; qui convertit `c` en minuscule (si cela est possible).

Exercice 3 (4pts) On dit qu'un tel tableau `tab` d'entiers est une suite arithmétique s'il existe un entier `r` tel que `tab[i + 1] == tab[i] + r` lorsque `tab[i]` et `tab[i + 1]` sont définis. L'entier `r` est appelé la *raison* de `tab`. Par exemple, le tableau contenant successivement les valeurs 16, 13, 10, 7, 4, 1 est une suite arithmétique de raison `r = -3`. On veut écrire une fonction de prototype

```
int est_arithmetique(int tab[], int k);
```

qui renvoie 1 ou 0 selon que le tableau `tab`, de longueur `k`, contient ou non une suite arithmétique.

1. Écrire une version itérative de `est_arithmetique`.
2. Écrire une fonction *réursive*

```
int est_arithmetique_aux(int tab[], int k, int r);
```

qui renvoie 1 si le tableau `tab`, de longueur `k` est arithmétique de raison `r`. Utiliser cette fonction pour écrire une nouvelle version de la fonction `est_arithmetique`.