

*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 (2pts)** Répondre aux questions suivantes par vrai ou faux. 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. Vrai signifie « toujours vrai » et Faux signifie « pas toujours vrai », sauf lorsque la question évoque explicitement une possibilité.

Vrai    Faux

1. Les deux séquences suivantes

Séquence 1 : `static int x;`  
`x = 1;`

Séquence 2 : `static int x = 1;`

sont équivalentes si elles sont placées au début d'une définition de fonction. ....

2. Les fichiers manipulés par les fonctions de la bibliothèque standard sont terminés par un caractère spécial EOF.....
3. L'expression `f(i++,i++)` s'évalue comme `f(1,2)` lorsque `i` vaut initialement 1.....
4. Pour créer le fichier objet `essai.o` à partir du fichier source `essai.c`, la norme ANSI C 99 demande que toute variable utilisée dans `essai.c` soit déclarée avant utilisation.

**Exercice 2 (5pts)** 1. Écrire une fonction

`char *index (char *s, int c);`

qui renvoie un pointeur sur la première occurrence du caractère `c` dans la chaîne `s`.

2. Écrire une fonction

`char *strstr (char *s, char *motif);`

qui cherche la première occurrence de la sous-chaîne `motif` dans la chaîne `s`. Les caractères `'\0'` de fin de chaîne ne sont pas comparés. Elle renverra un pointeur sur le début de la sous-chaîne trouvée dans `s`, ou `NULL` si celle-ci n'est pas trouvée.

3. Écrire une fonction

`int strspn (char *s, char *accept);`

calculant la longueur du plus long segment initial de `s` qui ne contient que des caractères présents dans `accept`.

**Exercice 3 (13pts/20)** On considère des arbres binaires finis dont

- chaque nœud a 0 ou 2 fils;
- chaque nœud contient une valeur entière.

Un exemple est donné sur la Figure 1.

1. Définir un type `struct arbre` pour représenter de tels arbres.
2. Écrire une fonction

`int taille(struct arbre *A);`

qui calcule le nombre de nœuds de l'arbre pointé par `A`. Sur l'exemple de la Figure 1, la fonction `taille` doit renvoyer 5.

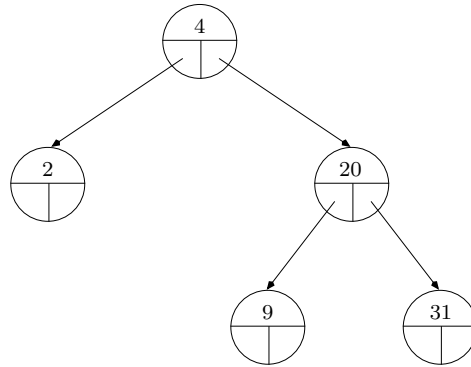


FIG. 1: Un arbre binaire dont les nœuds sont étiquetés par des entiers

3. Écrire une fonction

```
int hauteur(struct arbre *A);
```

qui renvoie la hauteur de l'arbre pointé par **A**, c'est-à-dire la longueur de sa plus longue branche. Sur l'exemple de la Figure 1, la fonction `hauteur` renverra 3 (les branches 4, 20, 31 et 4, 20, 9 ont cette longueur).

4. Écrire des fonctions

```
int max(struct arbre *A);
int min(struct arbre *A);
```

qui calculent le maximum et le minimum, respectivement, des valeurs se trouvant dans les nœuds de l'arbre pointé par **A**. Sur l'exemple de la Figure 1, la fonction `max` renverra 31 et la fonction `min` renverra 2.

5. On dit qu'un arbre est *trié* si, pour chaque nœud  $N$  étiqueté par  $x$  qui a 2 fils :

- tous les nœuds du sous-arbre gauche de  $N$  sont étiquetés par une valeur inférieure ou égale à  $x$ .
- tous les nœuds du sous-arbre droit de  $N$  sont étiquetés par une valeur supérieure à  $x$ .

Par exemple, l'arbre de la Figure 1 est trié. Écrire une fonction

```
int est_trié (struct arbre *A);
```

qui renvoie 1 si l'arbre pointé par **A** est trié et 0 sinon.