Cours d'introduction à l'informatique
Examen de janvier 2025

Livres fermés. Durée : 3 h 30.

Veuillez répondre aux questions sur des feuilles séparées sur lesquelles figurent votre nom, prénom et matricule. Tous les programmes et fragments de code demandés doivent être rédigés en langage C.

1. (a) Un tableau est dit *monotone* si ses éléments sont triés par ordre croissant ou décroissant. Par exemple, les tableaux $[2, 0, 0, -1]$, $[42]$ et $[10, 10, 20, 30]$ sont monotones, mais les tableaux $[23, 1, 2025]$ et $[-1, 1, 1, -1]$ ne le sont pas.

   Écrire une fonction qui prend en arguments un tableau d'entiers `t` et sa taille `n`, et qui retourne 1 si `t` est monotone et 0 sinon.

   (b) Par la méthode des invariants, démontrer que la valeur calculée par cette fonction est correcte.

2. (a) Écrire une procédure prenant en arguments un tableau d'entiers non signés `t`, sa taille `n`, un tableau `u`, et la taille `m` de ce dernier. On suppose que les éléments de `t` sont tous différents et que l'on a $\texttt{m} \leq \texttt{n}$. La procédure demandée doit écrire dans `u` les `m` plus petits éléments de `t`, dans un ordre quelconque. Le contenu de `t` doit rester inchangé.

   Par exemple, si `t` contient $[1, 5, 3, 0, 4, 7]$ et $\texttt{m} = 3$, alors la procédure doit écrire dans `u` les valeurs 0, 1 et 3.

   Note : On ne demande pas la solution la plus efficace possible ; un algorithme polynomial suffit.

   (b) Calculer la complexité en temps de la procédure obtenue au point (a).

3. (a) Décrire le plus simplement possible ce que calcule la fonction C suivante :

```
unsigned f(unsigned a, unsigned b)
{
  if (a < b || b == 0)
    return 0;

  return 1 + f(a - b, b);
}
```

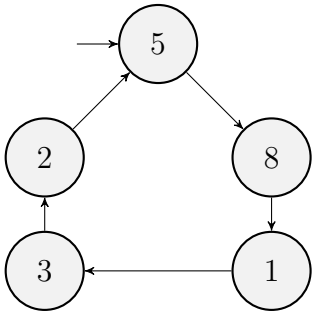
   (b) Quelle est la complexité en espace de cette fonction ? (Justifier votre réponse.)

   (c) Écrire une fonction réalisant exactement la même opération, mais sans effectuer d'appel récursif ni d'itération de boucle.

4. Un *graphe cyclique* de taille $n > 0$ est composé d'un ensemble de n *nœuds*, chacun d'entre eux étant caractérisé par un *identifiant* entier qui lui est propre et par un pointeur vers un nœud appelé son *successeur*. Un des nœuds du graphe est distingué comme étant *initial*.

   Pour être valide, le graphe doit former un unique cycle. Cela signifie que si on le parcourt en passant répétitivement d'un nœud à son successeur, en commençant à partir d'un nœud quelconque, on finira toujours par visiter tous les nœuds.

   Par exemple, la figure suivante montre un graphe cyclique de taille 5, dont les identifiants des nœuds forment l'ensemble $\{1, 2, 3, 5, 8\}$. Le successeur de chaque nœud est représenté par une flèche, et la flèche sans origine indique que le nœud initial est celui d'identifiant 5.



   (a) Écrire des fragments de code définissant deux types structurés permettant de représenter un nœud d'un graphe cyclique, et un graphe cyclique. On demande que chaque champ de ces types structurés corresponde soit à l'identifiant d'un nœud, soit à un pointeur vers un nœud.

   (b) Écrire une fonction prenant en arguments un tableau d'identifiants et la longueur de ce tableau, et retournant un pointeur vers une représentation nouvellement allouée d'un graphe cyclique dont les nœuds successifs correspondent aux éléments du tableau. Le premier élément du tableau définit le nœud initial. Par exemple, si la fonction reçoit en entrée le tableau $[5, 8, 1, 3, 2]$, alors elle doit produire le graphe cyclique de la figure précédente. Le type de la représentation générée doit bien sûr être compatible avec votre réponse au point (a).

   (c) Écrire une procédure acceptant en argument un pointeur vers une représentation d'un graphe cyclique, chargée d'afficher dans l'ordre l'identifiant de tous les nœuds visités lors d'un parcours de ce graphe. Par exemple, pour le graphe de la figure précédente, elle doit afficher "`5 8 1 3 2`". On demande également que cette procédure libère tout l'espace mémoire occupé par la représentation du graphe.

   Note : Vous pouvez programmer des fonctions ou des types de données supplémentaires si votre solution le nécessite.