Examen d'algorithmique

Samedi 17 octobre 2015 9h–11h / Aucun document autorisé

Mode d'emploi : Le barème est donné à titre indicatif. **La qualité de la rédaction des algorithmes et des explications sera très fortement prise en compte pour la note.** On peut toujours supposer une question résolue et passer à la suite.

Exercice 1 : Dérouler un algorithme itératif (3 points)

On considère l'algorithme ci-dessous :

```
Def P(entier x) :
tant que x != 1 faire:
   Afficher x
   Si x est pair Alors:  x = x/2
     Sinon:  x = 3*x+1
```

1. Décrire ce que fait l'algorithme `P` appelé avec le paramètre 3.
2. Décrire ce que fait l'algorithme `P` appelé avec le paramètre 22.

Exercice 2 : Dérouler un algorithme récursif (4 points)

On considère l'algorithme ci-dessous :

```
Def P(tableau T, entiers: bg , bd) :
    Si (bg==bd)  Alors :
         a1 = T[bg]
         a2 = T[bg]
      Sinon :
        m = (bg+bd)/2
        (x1,y1) = P(T,bg,m)
        (x2,y2) = P(T,m+1,bd)
        Si x1<x2 Alors:  a1 = x1
             Sinon:   a1 = x2
        Si y1>y2 Alors:  a2 = y1
             Sinon:    a2 = y2
Retourner (a1,a2)
```

1. Appliquer l'algorithme sur le tableau `[3,6,2,10,67,34]` avec `bg=0` et `bd=5` (on suppose que le premier indice du tableau est 0). On précisera tous les appels récursifs effectués et leurs résultats.
2. Que fait l'algorithme `P(T,0,n-1)` pour un tableau de taille `n` ?

Exercice 3 : Recherche dichotomique - 4 points

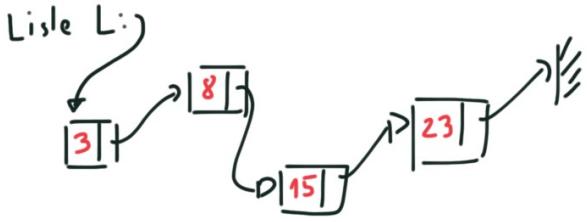
Ecrire un algorithme **itératif** de recherche dichotomique d'un entier `x` dans un tableau (d'entiers) `T` trié dans l'ordre croissant. L'algorithme retournera l'indice de `x` dans `T` si il est présent, et -1 sinon.

Exercice 4 : Listes chaînées - 6 points

On considère des listes chaînées d'entiers. Chaque cellule de la liste aura deux champs : un champ `val` pour stocker une valeur entière, et un champ `suiv` pour stocker l'adresse de la cellule suivante. Une liste est alors l'adresse de la première cellule (et 0 si c'est la liste vide). On utilisera les primitives classiques : `Alloc()` pour allouer de la mémoire pour y stocker une cellule (cette fonction retourne l'adresse de la zone mémoire choisie), et `c->val` et `c->suiv` pour accéder aux champs à une adresse `c`,...

On s'intéresse ici aux listes triées dans l'ordre croissant : on sait donc que le premier élément de la liste est le plus petit, le second est le deuxième plus petit, *etc.*

Par exemple, la liste suivante est correcte :



1. Donner un algorithme `Test(entier x, liste L)` qui renvoie vrai si il existe une cellule dans `L` contenant la valeur `x`, et faux sinon.
2. Donner un algorithme `Ajouter(entier x, liste L)` qui ajoute une cellule (au bon endroit pour que la liste reste triée) dans `L`, et qui renvoie la nouvelle liste (c-à-d. l'adresse de la première cellule de la liste).
3. Etant donné un tableau d'entiers `T` contenant n valeurs entières, donner un algorithme qui construit une liste chaînée triée avec toutes les valeurs de `T`.
4. Par rapport au tri par insertion dans un tableau, quel est l'intérêt d'utiliser une liste chaînée ?

Exercice (3 points) 

Vous avez 12 pièces, toutes de même poids exceptée une qui est légèrement plus lourde que les autres. Trouver la pièce "lourde" en 3 pesées seulement. Même problème avec 27 pièces.