LO21-2014A-FS01-01

A2014 Examen Final LO21 et LO27 Mercredi 14 Janvier 2015

Modalités :

- Durée : 2 heures
- Aucun document autorisé, ni machine à calculer, ni téléphone
- Le barème est donné à titre indicatif (±1)
- **Une feuille par exercice**

Exercice 1 (9 points) – Listes.

On considère un mot comme une liste de caractères sur l'alphabet {'a','b'}.

1) (3 points) Étant donnée une liste de caractères l composée uniquement de 'a' et de 'b', écrivez de manière itérative l'algorithme de la fonction *subfib(l: Liste): Liste* qui renvoie la liste de caractères obtenue en remplaçant dans la liste l chaque 'a' par "ab" et chaque 'b' par "a".
   Exemple : subfib("abaab") = "abaababa"
2) (3 points) Donner la version récursive de l'algorithme de la fonction *subfib(l: Liste): Liste.*
3) (3points) Écrivez de manière récursive l'algorithme de la fonction *fibo(n: Entier): Liste* qui renvoie le mot obtenu en appliquant n fois la substitution *subfib* au mot « a » (liste composée du caractère 'a').

En plus des fonctions définies en cours sur le type abstrait *Liste*, vous pouvez utiliser la fonction suivante:

- *concat* (l_1,l_2: *Liste*): *Liste* qui construit la liste résultat de la concaténation des listes l_1 et l_2 (les éléments de l_1 suivis des éléments de l_2 dans une même liste).

Exercice 2 (4 points) – Arbres binaires.

1) On souhaite écrire l'algorithme récursif du sous-programme *Parcours(a: Arbreb, l: Liste): Entier* qui parcourt l'arbre a en fonction de la valeur des éléments de la liste l (contenant uniquement des 0 et des 1).
   - En partant de la racine de l'arbre a, on lit les éléments de la liste l. Si l'élément lu est un 0 on descend sur le fils gauche de la racine, si c'est un 1 on descend sur le fils droit. On continue à descendre selon les mêmes règles jusqu'à ce que la liste soit vide. La fonction doit alors renvoyer la valeur du nœud de l'arbre sur lequel on se trouve (par exemple si la liste l contient 1, 0, 1 il faut renvoyer la valeur du fils droit du fils gauche du fils droit de la racine de l'arbre).
   - Dans le cas où l'arbre a est vide, *Parcours* doit retourner -1.
   - Dans le cas où la liste l est vide, *Parcours* doit retourner la valeur de la racine de l'arbre a.

Exercice 3 (7 points) – Liste de Listes.

On considère des listes de points triée par ordre croissant sur les abscisses. Rappelons qu'un point est caractérisé par un couple de réels : son abscisse x et son ordonnée y.

- (4 points) Écrire l'algorithme itératif du sous-programme *ListeAbscisse(lp: List): L* qui à partir d'une liste de points construit une liste où chaque élément est composé d'une abscisse et de la liste des ordonnées des points ayant cette abscisse.
  Exemple :
  Soit la liste des points LP = (1,2),(1,6),(3,7), (4,7),(4,6).(4,9)
  La liste à construire est L = (1,(2,6)), (3,(7)), (4,(7,6,9))
- (1 point) Donner en langage C la déclaration complète du type de la liste L.
- (2 points) Donner la traduction complète en langage C du sous-programme *ListeAbscisse(lp: List): L.*