Université BORDEAUX I | **Année 1993-94**
Session de septembre

Maîtrise d'informatique

Modèles et Techniques de Programmation (J.P.Braquelaire)

Durée: 2 heures
Tous documents autorisés, sauf les photocopies d'ouvrages

1 Première partie

On considère l'implémentation de la fonction d'élévation à la puissance

$$x\hat{\ }n : \ \mathbb{R}\times\mathbb{N}\longrightarrow\mathbb{R}$$
$$(x,n)x^n$$

au moyen de la fonction SCHEME suivante :

```
(define (x^n x n)
  (define (aux A x n)
    (cond
     ((= n 0) A)
     ((even? n) (aux A (* x x) (/ n 2)))
     (else (aux (* A x) x (- n 1)))))
  (if (= x 1) 1 (aux 1 x n)))
```

1) – 3 points – L'algorithme utilisé pour calculer la puissance est très efficace (le nombre de multiplications effectuées pour le calcul de x^n est le nombre de 1 de l'écriture de n, et le nombre d'additions est $\lfloor \log_2(n) \rfloor$). On veut pouvoir étendre ce programme à des calculs de puissance sur d'autres types d'objet: nombres, mots, matrices, polynômes... Pour cela on paramètre la fonction `aux` au moyen d'une liste `L` de 3 éléments :

   (*neutre*, *galit*, *produit*)

   dont le premier élément est un objet du type des objets manipulés et les deux suivants sont des fonctions sur ces objets. Cette liste est appelée l'*interprétation* de la fonction. Écrire la fonctionnelle `make-x^n` recevant une interprétation et construisant une fonction d'élévation à la puissance.

2) – 1 point – Comment obtient au moyen de `make-x^n` la fonction `x^n` ci-dessus ?

3) – 2 points – Soit `L` l'interprétation construite avec la liste d'éléments `()`, `eqv?` et `append`. Quel est le type des objets calculés avec cette interprétation ? Donner un exemple de construction de la fonction d'élévation à la puissance et de son utilisation.

2 Deuxième partie

Il s'agit dans cette partie de concevoir une implémentation objet en C ANSI d'un module de liste générique. Une liste est une suite ordonnée de références génériques. La liste possède un élément courant qui lorsqu'il est défini est un des éléments de la liste. L'élément courant peut être indéfini (c'est par exemple le cas avec une liste vide).

Le module possède les fonctionnalités suivantes :

- `new` : instanciation d'une liste vide
- `begin` et `end` : positionnement de l'élément courant au début et à la fin de la liste
- `forward` et `backward` : déplacement vers l'avant ou vers l'arrière de l'élément courant
- `insert` : insertion d'un élément à la suite de l'élément courant
- `get` et `set` : récupération et modification de la valeur de l'élément courant
- `delete` : libération d'une liste

1) – 3 points – Donner l'interface du module `liste` en indiquant l'effet de chaque fonction d'interface sur l'élément courant. Plusieurs choix sont possibles, mais le choix retenu doit permettre l'insertion en début de liste au moyen d'un enchaînement d'appels à des fonctions de l'interface.

2) – 3 points – Décrire au moyen d'une figure les structures de données utilisées dans l'implémentation.

3) – 3 points – Donner l'implémentation de la fonction d'insertion.

4) – 3 points – Ajouter à l'interface la fonction `search` de positionnement de l'élément courant par recherche de la première occurrence d'une valeur à partir de l'élément courant et décrire le principe de son implémentation.

5) – 2 points – Quel problème pose la libération d'une liste et comment le résoudre ?