TD
N° 7

LST–A (S3)
2009-2010
Algorithmes et Programmation Impérative 2

16 novembre 2009

1 Parcours de listes

1.- Parcours de listes

Exercice 1. *Somme des éléments d'une liste d'entiers*

Écrivez une fonction qui calcule la somme des éléments d'une liste d'entiers (`ELEMENT=CARDINAL`).

Exercice 2. *Affichage des éléments d'une liste dans l'ordre inverse*

Réalisez une procédure d'affichage des éléments d'une liste dans l'ordre inverse sans construire la liste inversée.

2 Modification de listes

2.- Modification de listes

Exercice 3. *Insertion dans une liste triée*

Réalisez une procédure d'insertion d'un élément e dans une liste triée l. La liste l a donc un élément de plus apès insertion et garde son caractère trié.

```
1 {l = (1, 2, 3, 4)}
2 inserer(3,l);
3 {l = (1, 2, 3, 3, 4)}
```

Exercice 4. *Suppression d'un élément dans une liste*

Question 4.1.

Réalisez la procédure `supprimeKeme` *qui supprime l'élément de rang k d'une liste l.*

```
1 // supprimer(k,l) modifie la liste l
2 // en supprimant son élément de rang k
3 // CU : 0 ≤ k < longueur(l)
4 procedure supprimerKeme(const k : CARDINAL;
5                         var l : LISTE);
```

Dans un premier temps, vous réaliserez cette procédure en ne libérant pas l'espace mémoire occupé par la cellule de l'élément supprimé. Vous modifierez ensuite la procédure afin de libérer cet espace mémoire.

Question 4.2. *Réalisez la procédure* `supprimePremiereOccur` *qui supprime la première occurrence d'un élément dans une liste.*

```
1 {l = (3, 1, 4, 1, 5, 9, 2)}
2 supprimePremiereOccur(1,l);
3 {l = (3, 4, 1, 5, 9, 2)}
```

Question 4.3. *Réalisez la procédure* `supprimeToutesOccur` *qui supprime toutes les occurrences d'un élément dans une liste.*

```
1 {l = (3, 1, 4, 1, 5, 9, 2)}
2 supprimeToutesOccur(1,l);
3 {l = (3, 4, 5, 9, 2)}
```

3 Construction de listes

2.- Construction de listes

Exercice 5. *Copie d'une liste*

Réalisez une fonction de copie de liste

Exercice 6. *Concaténation de deux listes*

La liste obtenue par concaténation de deux listes l_1 et l_2 est la liste constituée des éléments de l_1 suivis de ceux de l_2.

Par exemple, si $l_1 = (3, 1, 4)$ et $l_2 = (1, 5, 9, 2)$, alors la liste obtenue par concaténation de l_1 et l_2 est $l = (3, 1, 4, 1, 5, 9, 2)$

Question 6.1. *Réalisez une procédure à deux paramètres de type* `LISTE` *qui change la première liste en la concaténée*

```
1 {l1 = (3,1,4), l2 = (1,5,9,2)}
2 concatener(l1,l2);
3 {l1 = (3,1,4,1,5,9,2), l2 = (1,5,9,2)}}
```

Question 6.2. *Réalisez une fonction à deux paramètres qui construit une nouvelle liste obtenue par concaténation des deux listes passées en paramètre.*

```
1 {l1 = (3,1,4), l2 = (1,5,9,2)}
2 l := concatene(l1,l2);
3 {l1 = (3,1,4), l2 = (1,5,9,2)},
4  l = (3,1,4,1,5,9,2)}
```

Exercice 7. *Miroir d'une liste*

Le miroir d'une liste est une liste dont les éléments sont dans l'ordre inverse. Par exemple, le miroir de la liste $(3, 1, 4, 1, 5, 9, 2)$ est la liste $(2, 9, 5, 1, 4, 1, 3)$.

Réalisez une fonction `miroir` qui construit la liste miroir d'une liste passée en paramètre.

Exercice 8. *Liste de tous les éléments d'une liste staisfaisant un critère*

On considère qu'un prédicat nommé `critere` est défini.

```
1 function critere(e : ELEMENT) : BOOLEAN;
```

Réalisez une fonction qui construit la liste de tous les éléments d'une liste passée en paramètre satisafaisant le critère décrit par le prédicat `critere`.

Exercice 9. *Génération de listes d'entiers*

Question 9.1. *Réalisez une fonction qui construit la liste de tous les entiers de 0 à n, n étant le paramètre de la fonction*

1. *dans l'ordre décroissant d'abord ;*
2. *dans l'ordre croissant ensuite.*

Question 9.2. *Réalisez une fonction qui construit une liste de longueur l passée en paramètre, dont les éléments sont des entiers choisis au hasard entre 0 et n inclus, n étant passé en paramètre.*

Pour cela, vous pourrez utilisez la fonction

```
1 // random(a) = un entier compris entre 0 et a − 1
2 function random(a : CARDINAL) : CARDINAL;
```