TD
N° 6

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

2 novembre 2009

Exercice 1. *Utilisation des opérations primitives*

Question 1.1. *Après chacune des instructions ci-dessous, décrire les listes en énumérant leurs éléments.*

1. `l1 := LISTEVIDE ;`
2. `l1 := ajouteEnTete(1,l1)`
3. `l1 := ajouteEnTete(2,l1)`
4. `l1 := ajouteEnTete(3,l1)`

Question 1.2.

Trouver une unique instruction permettant de construire la liste l1 obtenue après la quatrième instruction ci-dessus.

Question 1.3. *Avec la liste l1 obtenue ci-dessus, que valent les expressions*

1. `tete(l1)`
2. `reste(l1)`
3. `tete(reste(l1))`
4. `reste(reste(l1))`
5. `tete(reste(reste(l1)))`
6. `reste(tete(reste(l1)))`

Question 1.4. *De manière générale, si* `l` *est une liste quelconque et* `x` *un élément quelconque, que donnent les expressions*

1. `tete(ajouteEnTete(x,l))`
2. `reste(ajouteEnTete(x,l))`

Question 1.5. *De manière générale, si* `l` *est une liste quelconque, que donne l'expression*

1. `ajouteEnTete(tete(l),reste(l))`

À quelle condition est-elle définie ?

Question 1.6. *Avec la liste* `l1` *construite dans la première question, indiquez les valeurs des listes obtenues après les instructions suivantes.*

1. `modifierTete(l1,5);`
2. `l2 := ajouteEnTete(6,reste(l1));`
3. `modifierReste(l1,l2);`
4. `modifiertete(l2,7);`

Question 1.7.

Que donne l'instruction `modifierReste(l1,l1)` *?*

Exercice 2. *Une procédure*

On suppose définie dans l'unité `U_Liste` définissant le type `LISTE` et les opérations primitives, dont voici le début

```
unit U_Liste ;

interface
uses U_Element;

const
   LISTEVIDE = NIL;
type
   LISTE = ^CELLULE;
   CELLULE = record
      info : ELEMENT;
      suivant : LISTE;
   end {CELLULE};
```

la procédure `exercice` dont voici l'implémentation

```
   procedure exercice(const l : LISTE);
   var
      l1   : LISTE;
      vide : BOOLEAN;
   begin
      write('(');
      l1 := l;
      vide := estListeVide(l1);
      while not(vide) do
      begin
         afficherElement(tete(l1));
         l1 := reste(l1);
         vide := estListeVide(l1);
         if not(vide) then  write(',');
      end {while};
      write(')');
   end {exercice};
```

dans laquelle la procédure `afficherElement`, définie dans l'unité `U_Element`, affiche la valeur de son paramètre.

Question 2.1.

Que font les appels `exercice(l)` à cette procédure ? Montrer les résultats obtenus avec les listes d'entiers

1. `l=()`
2. `l=(1)`
3. `l=(1,2,3)`

Question 2.2.

Donnez un algorithme récursif pour produire le même résultat.

Exercice 3. *Représentation contigüe des listes*

Question 3.1. *Déclarez en* PASCAL *un type LISTE avec la représentation contigüe suggérée en cours, pour des listes d'entiers.*

Avec cette déclaration du type LISTE, donnez la représentation de la liste vide, puis de la liste (3,1,4,1).

Question 3.2.

Réalisez les opérations primitives avec cette représentation, et pour chacune d'elles estimez son coût.