

## INF202 ASD

### Devoir surveillé du 25 novembre 2006

Durée : 1h20. Aucun document autorisé. Barème envisagé : 3+3+3+11

Les algorithmes qu'on vous demande d'écrire doivent utiliser le langage EXALGO défini en cours. Dans les exercices manipulant de listes chaînées d'entiers, ces dernières seront définies de la façon suivante :

```
type maillon_d_entier=  
  enreg  
    Info : entier;  
    Sv : ^maillon_d_entier;  
  finenreg;  
  
  liste_d_entiers = ^maillon_d_entier;
```

#### Exercice 1.1

Donnez deux versions, l'une itérative, l'autre récursive, de la fonction qui prend comme paramètre une liste chaînée de nombres entiers positifs et retourne leur somme.

#### Exercice 1.2

On considère le type *Pile d'entiers*, où les éléments susceptibles d'être empilés sont des entiers. On peut implémenter ce type abstrait en utilisant, comme unique structure de données, une liste simplement chaînée. Adapter les cinq primitives des piles à cette structure (on pourra utiliser des fonctions de manipulation de listes vues en TD sans avoir à les réécrire).

#### Exercice 1.3

On suppose d'avoir à simuler un éditeur de texte ayant un caractère spécifique pour l'effacement de caractère, la touche `backspace`, qu'on notera par exemple `#`. L'éditeur dispose aussi d'une touche d'effacement de tous les caractères à gauche dans la ligne, que l'on notera `$`.

On suppose disposer :

- du type `Ligne` défini auparavant comme une suite de caractères
- d'une fonction `FinDeLigne` pour détecter la fin d'une ligne
- d'une procédure `lire` pour extraire dans une ligne le premier caractère pas encore lu

Appliquée à la ligne de texte : *vingt-cinq\$25 novemm#bre 199###2006* la procédure imprimera le texte : *25 novembre 2006*.

La procédure suivante traite une ligne de caractères à l'aide d'une pile, en lisant un caractère à la fois.

```
1  procédure Edition_ligne (val l: Ligne);
2  var   P: Pile_de_caractères;
3        c: caractère;
4  debut
5      .....
6      tantque ( non FinDeLigne(l) ) faire
7          lire(c,l);
8          selon que
9              c = '#' : .....
10             c = '$' : .....
11             autrement: .....
12         finselonque
13     fintantque
14     Imprimer_ligne(P)
15 fin
```

Compléter les lignes 5,9,10 et 11 avec les instructions opportunes.

Ecrire la procédure `Imprimer_ligne` qui prend comme paramètre la pile de caractères et affiche la ligne corrigée.

#### Exercice 1.4

Soit la procédure `mystere` suivante prenant comme paramètre une suite quelconque d'entiers stockée dans une liste chaînée :

```
1  procédure mystère ( réf L : liste_d_entiers );
2  var   p1, p2: liste_d_entiers ;
3  debut
4      tantque ( non listevide?(L) et L^.Info > 0 ) faire
5          p1 := L;
6          L := L^.Sv;
7          libérer(p1)
8      fintantque
9      si ( non listevide?(L) ) alors
10         p2 := L;
11         tantque (p2^.Sv <> NIL) faire
12             si ( (p2^.Sv)^.Info > 0 ) alors
13                 p1 := p2^.Sv;
14                 p2^.Sv := p1^.Sv;
15                 libérer(p1)
16             sinon
17                 p2 := p2^.Sv
18             finsi
19         fintantque
20     finsi
21 fin
```

1. Faire tourner la procédure `mystère` en prenant comme argument la liste contenant la suite d'entiers 4,7,-4,0,1. Donnez les détails de l'exécution.
2. Sans donner les détails de l'exécution, donnez maintenant les résultats pour la suite 1,5,9 et pour la suite -2,0,3,-5.
3. Que fait la procédure `mystère` dans le cas général ? (deux lignes d'explication au maximum sont suffisantes)
4. On pourrait substituer aux lignes [5,6,7] et [13,14,15] l'appel à une primitive du type abstrait `Liste`. Quelle est cette primitive ?
5. Donnez la complexité en temps de l'algorithme en utilisant la notation  $O$  et justifiez en une ligne votre réponse.
6. Si on veut prouver l'algorithme par la méthode des assertions on a besoin de définir des invariants de boucle.  
 La phrase suivante vous suggère l'invariant de la boucle `tantque` des lignes 11 à 19 :  
**“la sous-liste composée des maillons entre `L` et `p2` ne .....”**.  
 Complétez la phrase précédente, dessinez le diagramme de flot concernant uniquement les instructions de la boucle `tantque` des lignes 11 à 19 et ajoutez les assertions nécessaires.(On ne demande pas de prouver la terminaison de l'algorithme)
7. On suppose maintenant que la suite d'entiers à traiter est stockée dans un tableau et non plus dans une liste chaînée. Adapter l'algorithme précédent à cette structure de données en conservant la même complexité et sans utiliser de tableaux auxiliaires (suggestion : utiliser deux indices, l'un pour la case du tableau dont on examine la valeur, l'autre pour la case dans laquelle on va réécrire le prochain élément).