Algorithmique 1

DS 1

Documents **non** autorisés

Exercice 1.1

Soit u une suite finie, non vide, d'entiers, on dit que i est un minimun local pour u si

$$\forall j, 1 \leq j < i \quad \text{on a :} \quad u_j > u_i$$

Ainsi 1 est minimun local quelque soit la suite u. A titre d'exemple la suite suivante de taille 14 possède 5 minima locaux qui se trouvent aux rangs $1, 3, 8, 9, 13$;

$$12, 18, 7, 34, 7, 45, 9, 6, 4, 56, 4, 6, 2, 34$$

```
fonction minima(ref U:tableau[1..N] d'entier): entier;
var i, tmp, rest: entier;
debut
     rest = 1;
     i = 2;
     tmp = U[1];
     tant que (i <= N) faire
           si tmp > U[i]  alors
                               tmp = U[i] ;
                               rest = rest + 1;
           fsi;
           i = i + 1;
     ftq;
     retourner rest;
fin;
```

Ecrire une version récursive terminale de la fonction `minima`.

Exercice 1.2

Dans l'exercice suivant, on considère l'implémentation par un tableau du type liste simplement chaînée `listeSC` par le type `listeSC_Car` avec une gestion de l'espace de stockage comme défini en cours et traité en TD.
Ecrire une fonction qui, étant donnée un objet X, supprime toutes les occurrences multiples de X dans une liste L. En utilisant uniquement les primitives.

```
fonction supprimer_multiple_occurrence(ref L: listeSC_Car, val X: car) :vide;
```

suite au verso

Exercice 1.3
Dans l'exercice suivant, on considère une liste doublement chainée d'objet :

```
fonction mystere(ref L:liste d'objet; ref x:objet): booleen;
  debut
    debutListe(L);
    tant que !estFinListe(L) et valeur(L)!=x faire
      suivant(L);
    fintantque
    si (!estFinListe(L)) alors
        précédent(L);
        supprimerAprès(L)
      retourner(vrai)
    sinon
      retourner(faux)
    finsi
  fin
finfonction
```

1. Que fait la fonction mystere
2. Ne présente-t-elle pas un défaut ?
   Si oui proposer une correction.

ANNEXE

```
listeSC= liste de type_predefini;
```

défini en cours avec les primitives suivantes :

- Accès
  ```
  fonction valeur(val L:liste d'objet) : objet;
  fonction debutListe(ref L:liste d'objet) :vide;
  fonction suivant(ref L:liste d'objet) : vide;
  fonction listeVide(val L:liste d'objet) : booleen;
  fonction getCleListe(val L: liste d'objet) : curseur;
  ```
- Modification
  ```
  fonction creerListe(ref L:liste d'objet) : vide;
  fonction insererApres(ref L:liste d'objet;
        val x:objet;) : vide;
  fonction insererEnTete(ref L:liste d'objet
         val x:objet) : vide;
  fonction supprimerApres(ref L:liste d'objet) : vide;
  fonction supprimerEnTete(ref L:liste d'objet) : vide;
  fonction setCleListe(ref L: liste d'objet, val c:curseur) : vide;
  fonction detruireListe(ref L:liste d'objet) : vide;
  ```