Université Bordeaux 1 Licence Informatique 2008/2009

Algorithmique 1

DS 1

Documents **non** autorisés

NB : Dans toutes les fonctions qui sont demandées, il faudra utiliser uniquement les primitives (aucune référence à la structure). Ces primitives sont rappelées en annexe.

Exercice 1.1

Soit l'implémentation de `listeSC` par `listeSC_Car` avec une gestion dynamique de l'espace de stockage.

```
cellule=structure
                valeurElement:car;
                pointeurSuivant:^cellule;
             finstructure;

listeSC_car=structure
              premier:curseur;
              cle:curseur;
            finstructure
```

Ecrire une fonction itérative qui inverse les éléments d'une liste **sans créer des nouvelles cellules**.

Exercice 1.2

Dans l'exercice suivant, on considère le type `listeDC`, liste doublement chaînée, défini en cours pour implémenter une suite d'entier.

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 $1, 3, 8, 9, 13$;

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

1. Écrire une fonction qui à partir d'une suite de taille fixe calcule le nombre de ses minima locaux.
2. Quelle est sa compléxité ?
3. Et si la liste était `listeSC`, quelle serait sa compléxité.

Exercice 1.3

```
fonction mystere(val n: entier)
var  i, l, k: entier;
     P, P2 : Pile d'entier;
debut
     creerPile(P);
     creerPile(P2);
     Pour i=1 a n faire
          l = lire();
          Tant que (non(pileVide(P)) et valeur(P)<l) faire
                k = valeur(P));
                depiler(P);
                empiler(P2,k);
          fin_tq;
          empiler(P,l);
          Tant que (non(pileVide(P2))  faire
                k = valeur(P2));
                depiler(P2);
                empiler(P,k);
          fin_tq;
     Fait;
     Tant que (non(pileVide(P))) faire
          ecrire(valeur(P));
          depiler(P);
     fin_tq;
fin;
```

La fonction `lire()` charge en mémoire l'entier tapé au clavier et la fonction `ecrire(x)` affiche à l'écran la valeur de x.

1. quelle suite d'entiers sera écrite sur la sortie si à l'entrée on a la suite 4 3 1 2 et on fait l'appel `Mystere(4)` ?

2. Si la suite $p_1, p_2, ..., p_n$ est telle que $p_1 > p_2 > ... > p_n$ quel est le contenu de piles `P` et `P2` à la fin de la première boucle `Tant que` pour `i = k , 1<k<n` par l'appel `Mystere(n)` ?

3. Si la suite $p_1, p_2, ..., p_n$ est telle que $p_1 < p_2 < ... < p_n$ quel est le contenu de piles `P` et `P2` à la fin de la première boucle `Tant que` pour `i = k , 1<k<n` par l'appel `Mystere(n)` ?

4. A votre avis à quoi sert la deuxième pile ? Quelle méthode (cachée) implémente la fonction `Mystere` ?

ANNEXE

Les primitives de `listeSC`

```
fonction valeur(val L:liste d'objet):objet;
fonction debutListe(ref L:liste d'objet);
fonction suivant(ref L:liste d'objet);
fonction listeVide(val L:liste d'objet): booleen;
fonction getCleListe(val L: liste d'objet):curseur;
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;
```

Plus les deux primitives de `listeDC`

```
fonction finListe(val L:liste d'objet):vide;
fonction precedent(val L:liste d'objet): vide;
```