Université Bordeaux 1 Licence Informatique 2011-2012

Algorithmique 1
Contrôle continu 1

Exercice 1 *Question de cours*
Donner la définition d'un algorithme récursif.
Proposer un algorithme récursif qui calcule la factorielle. Quelle est sa complexité (justifier brièvement) ?
Cet algorithme présente-t-il une récursivité terminale ? Pourquoi ?

Exercice 2 *Listes simplement chaînées*
Dans cet exercice on considère le type abstrait `listeSC_car` avec les primitives données en annexe A. On dit qu'une liste $L1$ est préfixe d'une liste $L2$ si la liste $L2$ commence par la liste $L1$. Par exemple la liste $L1 = (2, 4)$ est préfixe de la liste $L2 = (2, 4, 6, 7, 9)$.

1. Ecrire une fonction qui teste si $L1$ est un préfixe de $L2$.
2. Ecrire une fonction qui calcule la liste $L2$ privée de la liste $L1$ (dans l'exemple cela donnera $(6, 7, 9)$).

Exercice 3 *Listes doublement chaînées*
Dans cette exercice on considère le type abstrait des listes doublement chaînées `ListeDC_car`, implémentées par allocation dynamique (voir annexe B).

1. Ecrire la primitive `Inserer_En_Fin` qui insère le caractère X à la fin de la liste L.
2. On considère la fonction suivante.

```
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
```

   Que fait cette fonction ? Présente-t-elle un défaut ? Si oui, proposer une correction.

Annexe A **Liste simplement chaînée**

```
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;
  ```
- **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 detruireListe(ref L:liste d'objet) : vide;
  ```

Annexe B **Implémentation du type listeDC, liste doublement chaînée, par allocation dynamique**

```
curseur= ^cellule
car= type_predefini
cellule= structure
           valeurElement: car;
           pointeurPrecedent: curseur;
           pointeurSuivant: curseur;
         finstructure;

listeDC_car= structure
               premier: curseur;
               dernier: curseur;
               cle: curseur;
             finstructure;
```

La liste vide est représentée par `NIL`

- **Gestion de la mémoire**
  ```
  fonction new(ref p: ^cellule) : vide;
  fonction delete(ref p: ^cellule) : vide;
  ```
- **Accès**
  ```
  fonction valeur(val L:liste d'objet) : objet;
  fonction debutListe(ref L:liste d'objet) :vide;
  fonction finListe(ref L:liste d'objet) :vide;
  fonction suivant(ref L:liste d'objet) : vide;
  fonction precedent(ref L:liste d'objet) : vide;
  fonction listeVide(val L:liste d'objet) : booleen;
  fonction estFinListe(val L:liste d'objet) : booleen;
  ```

– **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 detruireListe(ref L:liste d'objet) : vide;
```