Algorithmique 1 : Devoir Surveillé 1

Listes

Durée : 30mn
Sans documents

Exercice 1.1 *Listes simplement chaînées*
Dans l'exercice suivant on considera le type `listeSC` de , liste simplement chaînée,

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

défini en cours avec les primitives suivantes :

```
fonction premier(val L: listeSC): ^type_predefini;
fonction suivant(val L: listeSC, val P: ^type_predefini): ^type_predefini;
fonction listeVide(val L: listeSC): booleen;
fonction creerListe(ref L: listeSC): vide;
fonction insererApres(val X: type_predefini, ref L: listeSC,
                                              val P: ^type_predefini): vide;
fonction insererEnTete(val X: type_predefini, ref L: listeSC): vide;
fonction supprimerApres(ref L: listeSC, val P: ^type_predefini): vide;
fonction supprimerEnTete(ref L: listeSC): vide;
```

1. Écrire une fonction `supprimer_doublons_sc` qui enlève les doublons dans une liste donnée

   ```
   fonction supprimer_doublons_sc(ref L: listeSC): vide;
   ```

Exercice 1.2 *Listes doublement chaîneées*

Dans l'exercice suivant on considera le type `listeDC`, liste doublement chaînée, défini en cours, les primitives étant les mêmes que pour le type `listeSC` en plus de :

```
fonction dernier(val L: listeDC): ^type_predefini;
fonction precedent(val L: listeDC, val P: ^type_predefini): ^type_predefini;
```

1. Écrire une fonction `supprimer_doublons_dc` qui enlève les doublons dans une liste donnée.

   ```
   fonction supprimer_doublons_dc(ref L: listeDC): vide;
   ```

2. Écrire une fonction `union` qui à partir de deux listes `L1` et `L2` données produit une nouvelle liste `L3` qui contient les éléments communs à `L1` et `L2`.

   ```
   fonction union(val L1: listeDC, val L2: listeDC, ref L3: listeDC): vide;
   ```