Enseignant : Moez Hammami

Série 2 : Listes simplement chainées

CORRIGE

Exercice 1

En considérant la définition d'une Liste simplement chainée suivante

```
type
 Liste=^noeud
noeud=enregistrement
            val :<type>
            suiv : Liste
fenreg
```

Donner l'implémentation de chacune des fonctions suivantes :

1. **ListeCopier(L : Liste) :Liste** qui permet de retourner une copie de la liste donnée comme argument.
2. **ListeComparer(L1 :Liste, L2 :Liste) :booléen** qui permet de retourner vrai si les deux listes ont les mêmes éléments dans le même ordre et de retourner faux dans le cas contraire. On supposera que la fonction taille(L) est définie.
3. **ListePurger(var L :Liste)** Cette fonction permet de purger une liste(supprimer les doublons).
4. **ListeRemplacer(var L :Liste, a :<type>, b :<type>) :entier** Cette fonction permet de remplacer toutes les occurrences d'un élément **a** par l'élément **b** et renvoyer le nombre de remplacements.
5. **Concatener(L1 :Liste, L2 :Liste) :Liste** Cette fonction permet de retourner une nouvelle liste contenant les éléments de la liste L1 suivis par les éléments de la liste L2 (en gardant le même ordre des éléments).

Correction

ListeCopier(L : Liste) :Liste

```
Var
listeCopie :Liste
courant :Liste
newNoeud :Liste
Début
listeCopie :=NIL  //initialiser la nouvelle liste
i :=1 ;
Tant que (L<>Nil) faire
        Allouer(newNoeud)
        newNoeud^.val :=L^.val
        newNeud^.suiv :=NIL
       Si (i=1) alors
        listeCopie :=newNoeud
        courant :=newNoeud
      sinon
        courant^.suiv :=newNoeud
        courant :=courant^.suiv
     finsi
       L := L^.suiv
       i :=i+1
 fTantque
retourner listeCopie
Fin
```

```
ListeComparer(L1 :Liste, L2 :Liste) :booléen
Var
  egale:booléen
  courant1 :Liste ; courant2 :Liste
Début
   Egale :=vrai
   si (Taille(L1) <>Taille(L2)) alors
         retourner faux
   sinon
      si (L1=NIL et L2=NIL) alors
           retourner Vrai
      sinon //les deux listes sont de même taille et ils ne sont pas vides
        Tant que (courant1 <>NIL et courant2<>NIL et egale=vrai) faire
          Si (courant1^.val <>courant 2^.val) alors
                egale :=faux
          Sinon
                Courant1 :=courant1^.suiv
                Courant2 :=courant2^.suiv
          Finsi
       finTantque
   finsi
finsi
retourner (egale)
fin
```

```
ListePurger(var L :Liste)
Var Noeudi :Liste, Noeudj :liste, precedent :Liste ;
//precedent est utilisé pour supprimer le nœudj
Début
  Noeudi :=L
 precedent :=L
 Tantque (Noeudi <>Nil) faire
     Noeudj := Noeudi^.suiv
     Tantque (Noeudj <> Nil) faire
       Si (Noeudj^.val = Noeudi^.val) alors
          //supprimer Noeudj
          precedent^.suiv :=Noeudj^.suiv.
          Libérer(Noeudj)
          Noeudj :=precedent^.suiv
       Sinon
         precedent :=Noeudj
         Noeudj :=Noeudj^.suiv
      Fin si
     fTantque
  Noeudi=Noeudi^.suiv
fTantque
fin


ListeRemplacer(var L :Liste, a :<type>, b :<type>) :entier
Var
   Var courant :Liste, nbreRemplace :int
Début
        courant := L
        nbreRemplace :=0
        tantque courant <> Nil faire
                si (courant^.val=a) alors
                        courant^.val :=b
                        nbreRemplace := nbreRemplace+1
                fin si
                courant := courant^ .suiv
        ftantque
   retourner nbreRemplace

Fin
```

```
Concatener(L1 :Liste, L2 :Liste) :Liste
Var listResultat :Liste, courant :Liste
     nouveau,precedent :liste , i :entier
Début
 listResultat :=Nil
 courant=L1 ;
i=0 ;
 tantque (courant <> NIL)
         i=i+1
         allouer(Nouveau)
         nouveau^.val := courant^.val
         nouveau^.suiv :=NIL
         Si (i = 1) alors
            listResultat :=nouveau
            precedent :=nouveau
        Sinon
          precedent^.suivant :=nouveau
          precedent :=nouveau
        finsi
        courant:= courant^.suiv
 fin tantque

courant=L2 ;
 tantque (courant <> NIL)
         i=i+1
         allouer(nouveau)
         nouveau^.val := courant^.val
         nouveau^.suiv :=NIL
         Si (i = 1) alors  //c'est-à-dire la première liste était vide
            listResultat :=nouveau
            precedent :=nouveau
        Sinon
         precedent^.suivant :=nouveau
         precedent :=nouveau
        finsi
        courant:= courant^.suiv
 fin tantque
 retourner(listResultat)
fin
```

Exercice 2

On se propose de modéliser la gestion des patients dans un cabinet médical. Un patient est caractérisé par: le nom, le prénom et un champ ***rdv*** (pour rendez-vous) de type entier indiquant si le patient a un rendez-vous ou pas: 0 si le patient est sans rendez-vous, 1 si la patient est avec rendez-vous.

Avant d'être consultés par le médecin, les patients sont entrés dans une salle d'attente qui sera modélisée par une liste simplement chaînée de patients. Une secrétaire fait entrer les patients ayant un ***rdv=1*** selon leur ordre d'arrivée, ensuite elle fait entrer les autres patients (ceux dont le ***rdv=0***) selon leur ordre d'arrivée aussi.

1. Définir le type ***Patient***.
2. Définir le type ***Nœud*** dans le cadre d'une liste simplement chainée de patients.
3. Définir le type ***Liste_patients*** pouvant contenir une liste de patients

```
type
Patient =enregistrement
        nom : chaine
        prenom : chaine
        rdv :entier
fenreg

noeud=enregistrement
  val : Patient
  suiv : Liste_patients
fenreg

Liste_patients =^noeud
```

4. Dans une deuxième étape, on vous demande d'écrire les Procédures suivantes :

4.1. ***AjoutPatient (var L :Liste_patients, nom :chaine, prenom :chaine, rdv :entier)*** , qui permet d'ajouter un nouveau patient à la fin de la liste L

Procédure AjoutPatient (var L :Liste_patients, nom :chaine, prenom :chaine, rdv :entier)

Var : courant :Liste_Patients
nouveau :Liste_Patients
P :Patient

Début

```
P.nom :=nom
P.prenon :=prenom
P.rdv :=rdv
Allouer(nouveau)
nouveau^.val :=P
nouveau^.suiv :=NIL
si (L=NIL) alors
   L :=nouveau
Sinon
  courant :=L
  Tant que (courant^.suiv <> NIL) faire
    courant :=courant^.suiv
  ftq
  courant^.suiv :=nouveau
fsi
```

Fin

4.2. ***Procédure RendezVous (L :Liste_patients, Var rdv :entier, Var sansRdv :entier)***, qui compte le nombre de patients avec rendez-vous (dans la variable rdv), et le nombre de patients sans rendez-vous (dans la variable sansRdv).

Procédure RendezVous (L :Liste_patients, Var rdv :entier, Var sansRdv :entier)
Var : courant :Liste_Patients

```
Début
        rdv :=0
        sansRdv :=0
        courant := L
        Tant que (courant !=NIL) faire
                Si (courant^.val.rdv=1) alors
                        rdv := rdv+1
                Sinon
                        sansRdv :=sansRdv+1
               finsi
          courant :=courant^.suiv
        ftq
Fin
```

4.3. ***Procédure SupprimePatient (Var L :Liste_patients)*** , qui permet de faire entrer un patient en consultation. Cette opération est effectuée de la manière suivante : s'il n'y a aucun patient avec rendez-vous alors c'est le premier patient de la liste qui est supprimé. Sinon on supprime le premier patient qui a un rendez-vous.

Procédure SupprimePatient (Var L :Liste_patients)
Var courant :Liste_patients
nbRdv, nbSansRdv:entier
predecesseur:liste_Patients
Début

```
        //vérifier le nombre de clients ayant un rdv
        RendezVous ( L, nbRdv, nbSansRdv)
        Si (nbRdv=0 et nbSansRdv=0) alors
                Ecrire("il n y a plus de patients")
        Sinon
           Si (nbRdv=0)  alors
                //supprimer le premier patient
                Ecrire ("Patient à servir: ", L^.val.nom, " ", L^.val.prenom)
                courant:=L;
```

```
            L:=L^.suiv
            Libérer(courant)
        Sinon // il y' au moins un patient avec rdv
          Si (L^.val.rdv=1) alors //cas de suppression en tête
            Ecrire ("Patient à servir: ", L^.val.nom, " ", L^.val.prenom)
            courant:=L;
            L:=L^.suiv
            Libérer(courant)
          Sinon // chercher le premier patient ayant un rdv
            courant := L
            Tant que (courant^.val.rdv=0)
                Predeccesseur:=courant
                courant=courant^.suiv
            fTantque
            //courant pointe sur le client à supprimer
            // et prédécesseur pointe sur le nœud qui le précède
            Ecrire ("Patient à servir: ", courant^.val.nom, " ",
             courant^.val.prenom)
            predecesseur^.suiv=courant^.suiv
            libérer(courant)
          finsi
        fsi
    fsi

Fin
```

4.4 ***Procédure ConsulterSalleAttente (L :Liste_patients)*** , qui affiche tout d'abord les Patients avec rendez-vous, ensuite les patients sans rendez-vous.

```
Procédure ConsulterSalleAttente (L :Liste_patients )
Var courant :Liste_patients
      i:entier
Début
courant :=L
i :=0 ;
Tant que  (courant <> NIL) faire
   Si (courant^.val.rdv=1) alors
```

```
            i :=i+1
            Ecrire( " patient avec rdv num  ", i ," " ,courant^.val.nom, "   " ,
            courant^.val.prenom)
        fsi
           courant :=courant^.suiv
    FTantque
   Courant := L
   i=0
   Tant que  (courant <> NIL) faire
      Si (courant^.val.rdv=0) alors
           i :=i+1
          Ecrire( «" patient sans rdv num ",i, " " ,courant^.val.nom, "   " ,
             courant^.val.prenom)
       fsi
             courant :=courant^.suiv
    FTantque
fin
```