Correction TD listes chainées

Fonction CREER( ): ↑Maillon
Var tête : ↑Maillon
Début
tête ← nouveau (Maillon)
Ecrire (donner une valeur entière:)
Lire (tête ↑.val)
tête →suivant ←NIL
Renvoyer (tête)
Fin
Fonction EST_VIDE (tete : ↑Maillon) : booléen
Si (L = NIL) Alors Renvoyer VRAI
Sinon Renvoyer FAUX
Fin Si
Fin
Procedure CONSTRUIRE(E/S tete : ↑Maillon)
Var element, courant : ↑Maillon
Rep : chaine
Début
Nouveau(element)
Lire(element↑.val)
tele →suivant ←element
element→suivant ←nil
courant ←element
Ecrire « voulez vous rajouter un nouvel élément : »
Lire rep
Tant rep = « oui » faire
Nouveau(element)
Lire(element↑.val)
courant →suivant ←element
element→suivant ←nil
courant ← element
Ecrire « voulez vous rajouter un nouvel élément : »
Lire rep
Fintantque
Fin
Procédure affichage_parcours_iter(E/S tete : ↑cellule**)**
var courant : ↑cellule
début
courant ←tete
tant que courant ≠ nil faire
écrire(courant↑.val)
courant ← courant↑.suiv
fin tant que
fin
Procédure affichage_parcours_recurs(E/S tete : ↑cellule**)**
var courant : ↑cellule
début
courant ←tete
tant que courant ≠ nil faire
écrire(courant↑.val)

affichage_parcours_recurs(courant↑.suiv)
fin tant que
fin

fonction longueur (tete : ↑Maillon) : entier
Var courant : ↑Maillon
Cpt : entier
Début
courant ← tete
tant que courant < > NIL
cpt ←cpt +1
ecrire « courant ↑. Val)
courant ← courant→suivant
fintantque
renvoyer cpt
fin
fonction Premier(tete : ↑Maillon) : entier
début
si tete <>nil alors renvoyer tete↑.val
sinon renvoyer null
finsi
fin
fonction dernier(tete : ↑Maillon) : entier
Var courant : ↑Maillon
Début
courant ← tete
répeter
courant ← courant →suivant
jusqu'à courant = nil
renvoyer courant↑.val
finsi
fin
fonction APPARTENIR(tete : ↑Maillon, x : entier) :booleen
var courant : ↑Maillon
B : booleen
début
si tete=NIL alors renvoyer faux
sinon courant ← tete
b ←faux
répéter
si courant↑.val = x alors B←vrai
sinon B ←faux
finsi
courant ← courant → suivant
jusqu'à B= vrai ou courant = NIL
renvoyer B
finsi
fin
fonction RECHERCHER (tete : ↑Maillon, x : entier):entier
var courant : ↑Maillon
cpt : entier

```
B : booleen
début
si tete=NIL alors renvoyer NIL
sinon courant ← tete
b ←faux
répéter
si courant↑.val = x alors B←vrai
                          cpt ←courant
sinon B ←faux
finsi
courant ← courant → suivant
jusqu'à B= vrai ou courant = NIL
renvoyer cpt
finsi
fin
 fonction Rechercher_recursive(x : entier ; tete : ↑Maillon ) : booleen
var courant : ↑Maillon
début
si tete = nil alors renvoyer faux
sinon
courant ← tete
si courant.val = x alors renvoyer vrai
sinon renvoyer (x, courant →suivant)
finsi
finsi
fin
procedure inserer_debut(E/S tete : ↑Maillon ; x : entier)
var element : ↑Maillon
debut
element ←nouveau(maillon)
element↑.valeur ←x
si tete = NIL alors tete ← element
                tete →suivant ←NIL
sinon element →suivant ←tete
finsi
fin
procedure inserer_fin(E/S tete : ↑Maillon  ; x : entier)
var element, courant : ↑Maillon
debut
element ←nouveau(maillon)
element↑.valeur ←x
si tete = NIL alors tete ← element
                 tete →suivant ←NIL
sinon courant ←tete
tant que courant <>nil faire
courant ←courant →suivant
fintantque
courant →suivant ←element
element →suivant ←NIL
finsi
fin
```

procedure inserer_avant(E/S tete : ↑Maillon **; E x : entier ; E pos : entier)**
var element, precdent, courant : ↑Maillon
debut
nouveau(element)
element↑.val ←x
precedent ←tete
courant ←tete →suivant
tant que courant <>pos faire
precedent ←precedent →suivant
courant ←courant →suivant
fintant que
fin
procedure supprimer_debut(E/S tete : ↑Maillon **)**
var courant: ↑Maillon
debut
si tete = NIL alors écrire « liste vide »
sinon courant ← tete
tete ←tete →suivant
libérer(courant)
finsi
fin

procedure supprimer_fin(E/S tete : ↑Maillon **)**
var courant, precedent: ↑Maillon
debut
si tete = NIL alors écrire « liste vide »
sinon
courant ←tete →suivant
precedent ←tete
tant que courant <>nil faire
courant ←courant →suivant
précdent ← precedent →suivant
fintantque
precdent →suivant ←nil
libérer (courant)
finsi
fin

procedure supprimer_courant(E/S tete : ↑Maillon **)**
var courant, precedent: ↑Maillon
debut
precedent →suivant ←courant →suivant
libérer (courant)
courant ←precdent
finsi
fin

procedure vidage(E/S tete : ↑**Maillon)**
var courant, precdent : ↑Maillon
début
Si tete = nil alors ecrire liste vide

Sinon
Courant ←tete →suivant
Tant que courant <>nil faire
Precedent ← courant
Courant ← courant → suivant
Liberer(precedent)
Fintant que
Liberer(courant)
Finsi
Fin

Fonction Premier(tete : Maillon) : entier
debut
Renvoyer (tete.val)
Fin

Fonction fin(tete : maillon) :entier
Var courant : entier
Courant ←tete →suivant
Tant que courant <>nil faire
Courant ← courant →suivant
Fintantque
Renvoyer courant.val
Fin
Fonction occur(tete : maillon, x : entier) :entier
Var courant : maillon
C :entier
Debut
C ←0
Courant ←tete
Tantque courant <>nil faire
Si courant.val = x alors c ←c+1
Finsi
Courant ←courant →suivant
Fintanque
Renvoyer c
Fin

Exercice

Algorithme Construction Liste
Type structure Maillon
valeur : entier
suivant : ↑Maillon
Fin
Var : L : ↑ Maillon
Début
L.valeur ← 5
L.suivant ← NIL
L ← Cons(13, L)
L ← Cons(21, L)
L ← Cons(17, L)

L ← Cons(4, L)
Fin

Fonction Longueur Liste(L : ↑Element) : entier
Si (L = NIL) Alors Renvoyer 0
Sinon Renvoyer 1 + Longueur Liste(Queue(L))
Fin Si
Fin

Procédure Ajout Liste(VAR L : Element, X : entier)
L←Cons(X, L)
Fin

Procedure Retrait Liste(E/S L : ↑Element , S X : entier)
Si (L = NIL) Alors Ecrire ("La liste est vide !")
Sinon X←Tete(L)
L ← Queue(L)
Fin Si
Fin

Fonction Indice(L : ↑Element, n : entier) : entier
Si (L <> NIL) Alors
Si (n = 1) Alors Renvoyer Tete(L)
Sinon Renvoyer Indice(Queue(L), n - 1)
Fin Si
Fin Si
Fin

Fonction Appartient(x : entier, L : ↑Element) : booléen
Si (L = NIL) Alors Renvoyer FAUX
Sinon
Si (Tete(L) = x) Alors Renvoyer VRAI
Sinon Renvoyer Appartient(x, Queue(L))
Fin Si
Fin Si
Fin

Fonction Concat(L1 : ↑Element, L2 : ↑ Element) : ↑ Element
Si (L1 = NIL) Alors Renvoyer L2
Sinon
Renvoyer Cons(Tete(L1), Concat(Queue(L1),L2))
Fin Si
Fin

Fonction Insertion Triee(x : entier, L : ↑Element) : ↑ Element
Si (L = NIL) Alors Renvoyer Cons(x, Liste Vide())
Sinon
Si (Tete(L)>x) Alors Renvoyer Cons(x, L)
Sinon
Renvoyer Cons(Tete(L),Insertion Triee(x, Queue(L)))
Fin Si

Fin Si
Fin
Exercice 9
Fonction Fusion(L1 : ↑Element, L2 : ↑ Element) : ↑ Element
Debut
Si (L1 = NIL) Alors Renvoyer L2
Fin Si
Si (L2 = NIL) Alors Renvoyer L1 Fin Si
Si (Tete(L1) > Tete(L2)) Alors
Renvoyer Cons(Tete(L1), Fusion(Queue(L1), L2))
Sinon Renvoyer Cons(Tete(L2), Fusion(L1, Queue(L2)))
Fin Si
Fin

Exercice 11
procedure concat1(E tete1, tete2 : ↑Maillon ; S tete3 : ↑Maillon)
début
courant3 : ↑Maillon
si tete1= nil et tete2=nil alors tete3 ←NIL
sinonsi tete1= Nil et tete2 <>nil alors tete3 ←tete2
sinonsi tete1<> Nil et tete2 =nil alors tete3 ←tete1
sinon
tete3 ←tete1
courant 3 ← tete3
tant que courant 3 <>NIL faire
courant3 ← courant3 →suivant
fintantque
courant 3 →suivant ←tete2
fintantque
fin
procedure concat2(E tete1, tete2 : ↑Maillon ; S tete3 : ↑Maillon)
début
element, courant1, courant2, courant3 : ↑Maillon
si tete1= nil et tete2=nil alors tete3 ←NIL
sinonsi tete1= Nil et tete2 <>nil alors tete3 ←tete2
sinonsi tete1<> Nil et tete2 =nil alors tete3 ←tete1
sinon
tete3 ←tete1
courant 3 ←tete1 →suivant
tant que courant1<>nil faire
nouveau(element)
element.val ←courant1.val
courant1 ←courant1 →suivant
element →suivant ←nil
courant3 →suivant ←element
courant3 ←element
fintantque
courant3 →suivant ←tete2
courant 3 ← tete2 → suivant
courant2 ← courant3
tant que courant2<>nil faire

```
nouveau(element)
element.val ←courant2.val
courant2 ←courant2 →suivant
element →suivant ←nil
courant3 →suivant ←element
courant3 ←element
fintantque
fin
```

Exercice 12

procedure fusion_triee(E tete1, tete2 : ↑Maillon ; S tete3 : ↑Maillon)

```
début
element, courant1, courant2, courant3 : ↑Maillon
si tete1= nil et tete2=nil alors tete3 ←NIL
sinonsi tete1= Nil et tete2 <>nil alors tete3 ←tete2
sinonsi tete1<> Nil et tete2 =nil alors tete3 ←tete1
sinon
nouveau(tete3)
si tete1.val =< tete2 alors tete3.val ←tete1.val
                            tete3 →suivant ←nil
                            courant1 ←tete1→suivant
sinon tete3.val ←tete2.val
      tete3 →suivant ←nil
      courant2 ←tete2→suivant
finsi
finsi
finsi
finsi
courant3 ←tete3
finsi
tant que courant2<>Nil ET courant1 <>nil faire
nouveau(element)
si courant2.val =<courant1.val alors element.val ←courant2.val
                                    courant2 ←courant2 →suivant
sinon element.val ←courant1.val
      courant1 ←courant1 →suivant
finsi
element →suivant ←nil
courant3 →suivant ←element
courant3 ←element
fintantque

tant que courant1<>nil faire
nouveau(element)
element.val ←courant1.val
courant1 ←courant1 →suivant
element →suivant ←nil
courant3 →suivant ←element
courant3 ←element
fintantque
tant que courant2<>nil faire
```

```
nouveau(element)
element.val ←courant2.val
courant2 ←courant2 →suivant
element →suivant ←nil
courant3 →suivant ←element
courant3 ←element
fintantque
fin
```

Exercice 13

Procedure eclater1(E tete1 : ↑Maillon ; S tete2,tete3 : ↑Maillon)

```
Var element, courant :  ↑Maillon
Debut
Si tete1 = nil alors tete2 ←NIL
                     Tete3 ←NIL
Sinon
Nouveau (tete2)
Nouveau (tete3)
Courant1 ←tete1
Si tete1.val >= 0 alors tete2.val ←tete1.val
                        Tete2 →suivant ←nil
                        Courant1 ←tete1 →suivant
                        Courant2 ←tete2
Sinon tete3.val ←tete1.val
      tete3 →suivant ←nil
      Courant1 ←tete1 →suivant
      Courant3 ←tete3
Finsi
Tant que courant1 <>NIL faire
Nouveau (element)
element.val ←courant1.val
element →suivant ←nil
 Si courant1 .val >=0 alors
                        Courant2 →suivant ←élément
                        Courant 2 ←element
Sinon  courant3 →suivant ←element
        Courant3 ←element
Finsi
Courant1 ←courant1 →suivant
Fintantque
```

Fin

Procedure eclater2(E tete1 : ↑Maillon ; S tete2,tete3 : ↑Maillon)

```
Var element, courant :  ↑Maillon
Debut
Si est_vide(tete1) alors tete2 ←NIL
                     Tete3 ←NIL
Sinon
Si tete1.val >= 0 alors inserer_debut(tete2, tete1.val)
                        Courant1 ←tete1 →suivant
                        Courant2 ←tete2
Sinon inserer_debut(tete3, tete1.val)
```

```
        Courant1 ←tete1 →suivant
        Courant3 ←tete3
Finsi

Courant1 ←tete1
Tant que courant1 <>NIL faire
Si courant1 .val >=0 alors  Inserer_fin(tete2, courant1.val)
Sinon  inserer_fin(tete3, courant1.val)
Finsi
Courant1 ←courant1 →suivant
Fintantque
Fin
```