Corrigé EMD1 Algo2 2007/2008

```
// on supposera (pour des raisons d'efficacité) que les différents tableaux sont des variables globales
// toutes les autres var sont locales à chaque module

Constr_sous_listes( N : entier )
debut
        // initialiser les tetes de liste à NIL
        pour c = 'a' , 'z'
           Tab_adr[c] := NIL;
        fp
        // insérer chaque mot au début de sa sous-liste associée
        // cela évite de parcourir chaque sous-liste jusqu'à la fin
        pour i = 1 , N
           c := T[i][1];
           Allouer(p);  Aff-val( p, T[i] ); Aff_adr( p, Tab_adr[c] );  Tab_adr[c] := p;
        fp
fin // Constr_sous_listes


Rech_mot( L:entier )
debut
        // utilisant un tableau de ptr temporaire pour avancer dans chaque sous-liste
        pour c = 'a' , 'z'
           temp[c] := Tab_adr[c];
        fp
        stop := FAUX;
        TQ  (Non stop)
           stop := VRAI;
           pour c = 'a' , 'z'
                si (temp[c] <> NIL)
                   si (Long( Valeur(temp[c]) ) = L)
                         ecrire( Valeur(temp[c]) )
                   fsi;
                   temp[c] := Suivant( temp[c]);
                   stop := FAUX
                fsi // (temp[c] <> NIL)
           fp
        FTQ
fin // Rech


Concat( var tete:ptr )
debut
        tete := NIL;
        pour c = 'a' , 'z'
           si ( Tab_adr[c] <> NIL )
                si ( tete = NIL )
                   tete := Tab_adr[c]
                sinon
                   Aff-adr( p , Tab_adr[c] );
                fsi;
```

```
                p := Tab_adr[c];
                TQ ( Suivant( p ) <> NIL )
                   p := Suivant( p );
                FTQ
             fsi // ( Tab_adr[c] <> NIL )
          fp; // c = 'a' , 'z'

          si ( tete <> NIL )  Aff-adr( p , tete ) fsi
fin  // Concat


Inserer( mot:chaine, var tete:ptr )
debut
          si ( tete = NIL )
             Allouer( tete ); Aff-val( tete, mot ); Aff-adr( tete, tete );
             Tab_adr[mot[1]] := tete
          sinon
             Allouer( p ); Aff-val( p, mot );
             si ( Tab[mot[1]] <> NIL )
                   // s'il existait des mot commençant par la meme lettre que mot:
                   // on insère le mot à la 2e pos, juste apres Tab_adr[mot[1]]
                   Aff-adr( p, suivant( Tab_adr[mot[1]] ) );
                   Aff-adr( Tab_adr[mot[1]], p );
             sinon
                   // s'il n'existait aucun mot commençant par la lettre mot[1]:
                   // on cherche le précédent (q) dans la liste circulaire, à partir de la case
                   // la plus proche de Tab_adr[mot[1]]
                   c := mot[1] – 1;
                   si ( c < 'a' )  c:= 'z' fsi;
                   TQ ( Tab_adr[c] = NIL )
                      c := mot[1] – 1;
                      si ( c < 'a' )  c:= 'z' fsi;
                   FTQ;
                   q := Tab_adr[c];
                   // q pointe le début de la sous-liste qui doit précéder le nouveau mot (p)
                   // on se deplace jusqu'au dernier maillon de cette sous-liste.
                   // rmq: c'est possible que la liste circulaire n'etait formée que d'une seule sous-liste
                   //         d'où le test : Suivant(q) <> Tab_adr[c] ci-dessous
                   TQ ( Valeur(q) = Valeur(Suivant(q))  ET  Suivant(q) <> Tab_adr[c] )
                      q := Suivant(q)
                   FTQ;
                   // insertion du nouveau maillon (p) juste après q
                   Aff-adr( p, Suivant(q) );
                   Aff-adr( q, p );
                   // mettre à jour Tab_adr
                   Tab_adr[mot[1]] := p
             fsi // ( Tab[mot[1]] <> NIL )
          fsi // ( tete = NIL )
fin // Inserer
```