```
(* Ce programme n'a pas été testé, il peut contenir quelques erreurs. *)
(*                          Merci de votre compréhension *)


const Max = 10 ;
const Nil =  0 ;
type cellule = 0..Max ;
type position = integer ;
type valeur = char ; (* type "E", char pour cet exemple *)
type liste = record
      tete : cellule ;
      val : array[1..Max] of valeur ;
      sv : array[1..Max] of cellule ;
      libre : cellule ;
     end ;

procedure InitListe(var l) ;
begin
  l.tete := Nil ;
  InitLibre(l) ;
end ;

function Longueur(l : liste) : position ;
var c, n : integer ;
begin
  c := l.tete ;
  n := 0 ;
  while c <> Nil do begin
    c := l.sv[c] ;
    n := n + 1 ;
  end ;
  Longueur := c ;
end ;

function IndicePosition(l : liste, p : position) : cellule ;
var c, n : integer ;
begin
   c := l.tete ;
   n := 0 ;
   while ((c <> Nil) and (n < p)) do begin
      c := l.sv[c] ;
      n := n + 1 ;
   end ;
   if (n < p) then writeln('Erreur : position absente')
   else Element := l.val[c] ;
end ;


function Element(l : liste, p : position) : valeur ;
var c, n : integer ;
begin
   Element := l.val[IndicePosition(l,p)] ;
end ;
```



```
procedure Inserer(var l : liste, p : position, v : valeur) ;
var new, prec : cellule ;   n : integer ;
begin
   new := newLibre(l) ;
   if new = Nil then writeln('ERREUR')
   else begin
      l.val[new] := v ;
      if p = 1 then begin  (* cas particulier de l'insertion en tête *)
         l.sv[new] := l.tete ;
         l.tete := new ;
      end
      else begin
         prec := IndicePosition(l,p-1) ;
         l.sv[new] := l.sv[prec] ;
         l.sv[prec] := new ;
      end ;
   end ;
end ;

procedure Supprimer(var l : liste, p : position) ;
var n : integer ;  prec : cellule ;
begin
   if p = 1 then begin   (* cas particulier de suppression en tête *)
      freeLibre(l.tete) ;
      l.tete := l.sv[l.tete] ;
   end
   else begin
      prec := IndicePosition(l,p-1) ;
      freeLibre(l.sv[prec]) ;
      l.sv[prec] := l.sv[l.sv[prec]] ;
      end ;
  end ;
end ;


(* Convention : on renvoie Nil s'il n'y a plus de cellule libre *)
function newLibre(var l : liste) : cellule ;
begin
   if l.libre = Nil then newLibre := Nil
   else begin
      newLibre := l.libre ;
      l.libre := l.suiv[l.libre] ;
   end ;
end ;

procedure freeLibre(var l : liste, c: cellule) ;
begin
   l.suiv[c] := l.libre ;
   l.libre := c ;
end ;

procedure InitLibre(var l :  liste) ;
var i :  integer ;
begin
   for i := 1 to Max-1 do
      l.suiv[i] := i+1 ;
   l.suiv[Max] := Nil ;
end ;


(* Il manque le programme principal *)
```