

Épreuve d'informatique de l'X - 2004 - MP/PC

Question 1 :

```
entier := proc(b :: nonnegint, c :: nonnegint)
  return(3*b+c)
end;
```

Question 2 :

```
poidsFort := proc(n :: nonnegint)
  return(n div 3) # en fait, return(iquo(n,3)) en Maple
end;

poidsFaible := proc(n :: nonnegint)
  return(n mod 3)
end;
```

Question 3 : Dans la fonction `longueurMotif` au niveau de la boucle `while`, on progresse tant que les caractères en position $i+k$ et $j+k$ sont identiques et que k ne dépasse pas m .

```
longueurMotif := proc(t :: array , i :: nonnegint ,
                     j :: nonnegint, m :: nonnegint)
  local k;
  k := 0;
  while t[i+k] = t[j+k] and k < m do
    k := k+1;
  od;
  return(k)
end;
```

Question 4 : On utilise évidemment la fonction précédente, en envisageant toutes les positions de départ ($i+k$) comprises entre i et $j-1$. Le but est de garder en mémoire la plus grande longueur de motif ce qui est fait à l'aide de la variable `l`.

(PSI*)

```
1  longueurMotifMax := proc(t :: array , i :: nonnegint,
2                           j :: nonnegint, m :: nonnegint)
3    local l, ll, k;
4    l := 0;
5    for k from 0 to min(m-1, j-i-1) do
6      ll := longueurMotif(t, i+k, j, m);
7      if ll > l then l := ll fi;
8    od;
9    return(l)
10   end;
```

Question 5 : Par rapport à la fonction «`longueurMotifMax`» précédente, on ajoute la déclaration des variables globales (ligne 3), la réinitialisation de ses mêmes variables et deux instructions fondamentales : mises à jour de `A` et de `L` ligne 9 puis de `C` ligne 12.

```
1  motifMax := proc(t :: array , i :: nonnegint,
2                  j :: nonnegint, m :: nonnegint)
3    local k, l;
4    global A, L, C;
5    L := 0; A := 0;
6    for k from 0 to min(m, j-i-1) do
7      l := longueurMotif(t, i+k, j, m);
8      if l > L then
9        A := k; L := l;
10     fi;
11   od;
12   C := t[j+L];
13   return(L)
14   end;
```

Question 6 : On décompose `A` et `B` en poids fort et poids faible et on imprime les entiers demandés à l'aide de la fonction d'impression formatée «`printf`».

```
1  imprimerTriplet := proc(A :: nonnegint, L :: nonnegint,
2                          C :: character)
3    printf("%1d%1d%1d%1d%c ", poidsFort(A), poidsFaible(A),
4          poidsFort(L), poidsFaible(L), C);
5  end;
```

Question 7 : Dans cette version de la fonction `compresser`, on suppose que la chaîne de caractères à compresser se termine par un chiffre supérieur ou égal à 3. Le principe de la fonction est le suivant : tant que le caractère de fin n'a pas été lu, on applique la fonction `motifMax` qui nous donne la longueur du plus grand motif qui se répète dans la fenêtre. Le motif doit démarrer dans la première moitié de la fenêtre et son sosie à partir de la position 9 de la fenêtre. Ensuite on fait glisser la fenêtre (ligne 8) en modifiant `i`. la variable `i` représente l'indice du début de la fenêtre et `i+9` son «centre».

```

1      compresseur := proc(s::array)
2      local i;
3      global A, L, C;
4      i := 0; C := 0;
5      while C <= 2 do
6          motifMax(s, i, i + 9, 8);
7          imprimerTriplet(A, L, C);
8          i := i + L + 1;
9      end do;
10     return() ;
11 end proc;
```

Question 8 : Pour décompresser, on suit l'algorithme décrit dans l'énoncé. La boucle `while` de la ligne 34 est nécessaire lorsque le motif qui se répète démarre dans la première moitié de la fenêtre et finit dans la seconde. Cela correspond au cas où $A + L > 8$. La fonction «`recolle`» prend en entrée deux tableaux et retourne en sortie la concaténation de ses deux tableaux. «`extrait`» est une fonction permettant d'extraire le sous tableau commençant à l'indice i et de longueur l dans le tableau t .

```

1      recolte := proc(t1::array, t2::array)
2      local i, n1, n2, res;
3      n1 := longueur(t1); n2 := longueur(t2);
4      res := array(0 .. n1 + n2 - 1);
5      for i from 0 to n1 - 1 do res[i] := t1[i] end do;
6      for i from 0 to n2 - 1 do res[i + n1] := t2[i] end do;
7      return eval(res);
8  end proc;
```

```

11     extrait := proc(t::array, i::nonnegint, l::nonnegint)
12     local res, k;
13     res := array(0..l - 1);
14     for k from 0 to l - 1 do
15         res[k] := t[i + k]
16     end do;
17     return res;
18 end proc;

19
20 affiche := proc(a::array)
21 local i;
22     for i from 0 to longueur(a) - 1 do printf("%d", a[i]) end do;
23     printf("\n");
24 end proc;

25
26 decompresser := proc(tc :: array)
27 local res, repere, i, Lprime;
28 global A, L, C;
29     res := array(0..8, [0,0,0,0,0,0,0,0]); repere := 0; Lprime := 0;
30     for i from 0 to iquo(longueur(tc),5) - 1 do
31         A := entier(tc[i*5],tc[i*5+1]);
32         L := entier(tc[i*5+2],tc[i*5+3]);
33         C := tc[i*5+4];
34         while L <> 0 do
35             if (longueur(res)-(repere+A+L))<0
36                 then Lprime := longueur(res)-(repere+A);
37                 else Lprime := L
38             fi;
39             res := recolte(res,extrait(res,repere+A,Lprime));
40             repere := repere + Lprime;
41             L := L - Lprime;
42         od;
43         res := recolte(res,array(0..0,[C]));
44         repere := repere + 1;
45     od;
46     affiche(res)
47 end;
```