ÉCOLE POLYTECHNIQUE - MP/PC - ÉPREUVE FACULTATIVE D'INFORMATIQUE 2004

Corrigé rédigé par Alain Schauber - alain.schauber@prepas.org

Version de MAPLE : Classic Worksheet Maple 10.
Les opérations arithmétiques infixes **div** et **mod** évoquées en préambule seront traduits en MAPLE respectivement par "iquo" et "irem", qui sont des opérateurs préfixes.

Nombres ternaires

Question 1

```
> entier:=(b,c)->c+3*b;
```

Question 2

```
> poidsFort:= x -> iquo(x,3);
  poidsFaible:= x -> irem(x,3);
```

Textes ternaires

Question 3

La fonction **longueurMotif** effectue une boucle dans chaque tour de laquelle on compare l'un après l'autre les caractères de même position relative à **i** et **j** de **t**, et on incrémente **l** tant qu'il y a égalité. Cette boucle s'arrête lorsqu'on constate une différence entre deux caractères comparés ou lorsqu'on a atteint la longueur maximale **m**. Alors on retourne **l**. La fonction s'exécute donc en temps linéaire par rapport à **m**; comme **m** doit évidemment être inférieur ou égal à **N**-1-**j** (ce qu'on ne vérifie pas dans cette fonction à caractère modulaire), elle est linéaire par rapport à **N**.

```
> longueurMotif:=proc(t,i,j,m) local l;
    # on suppose ici sans le vérifier que m est choisi pour que
  j+m <= N-1
    l:=0; #initialisation de l
    while l < m and t[i+l] = t[j+l] do l:=l+1 od;
    l
  end;
```

Question 4

On exécute une boucle impérative qui met au tour **k** la longueur maximale **l** à jour par comparaison avec celle calculée en partant des cellules d'indices **i**+**k** et **j** grâce à la fonction **longueurMotif**. Cette boucle s'arrête au bout de **a** tours, où **a** = 1+min(**m**-1,**j**-**i**-1) est déterminé de manière à respecter les contraintes **i**+**k** < **j** Sa complexité temporelle est donc de l'ordre de **a**O(**N**), et comme **a** <= **N**, elle est donc en O($\mathbf{N}^2$).

```
> longueurMotifMax:=proc(t,i,j,m) local l,k;
    # on suppose toujours sans le vérifier que m est choisi pour
  que j+m <= N-1
    l:=0;
    for k from 0 to min(m-1,j-i-1) do
  l:=max(l,longueurMotif(t,i+k,j,m)) od;
    l
  end;
```

Question 5

Il suffit de conserver dans une variable locale **a** l'indice **k** courant de la longueur maximale courante **l** . Une variable locale **lc** contient la longueur courante, à fin de comparaison avec **l**, et pour éviter un deuxième appel à la fonction **longueurMotif**, dont le coût est linéaire.

```
> motifMax:=proc(t,i,j,m) local l,a,k,lc;
    global A,L,C;
    l:=0; a:=0;
    for k from 0 to min(m-1,j-i-1) do
      lc:=longueurMotif(t,i+k,j,m);
      if lc > l then l:=lc; a:=k fi;
    od;
    L:=l; A:=a; C:=t[j+l];
    RETURN() # seul un effet de bord est souhaité ici
  end;
```

Question 6

Pour éviter l'impression sur 5 lignes ou le recours à des fonctions de pretty-printing, on imprime la séquence des 5 chiffres ternaires:

```
> imprimerTriplet:=(a,b,c)->print(poidsFort(a),poidsFaible(a),poid
  sFort(b),poidsFaible(b),c);
```

Question 7

Pour permettre l'impression du texte compressé sur une seule ligne, on remplace la fonction **imprimerTriplet** par une fonction analogue **calculerTriplet**, qui se contente de calculer la séquence sans l'imprimer.

La fonction est essentiellement constituée d'une boucle réalisant le processus décrit dans l'énoncé : à chaque tour de boucle, on exécute la fonction **motifMax**, qui met à jour les variables globales **A**, **L** et **C**. On concatène alors le le triplet correspondant sous la forme du quintuplet ternaire calculé avec **calculerTriplet**, et on "recentre" la fenêtre, ce qui revient à recentrer sa première cellule, représentée par la variable locale **i**. La boucle s'arrête dès qu'on arrive en fin de texte, c'est-à-dire **C** = x. Cet événement est identifié en remarquant que x est le seul caractère du texte qui n'est pas un chiffre ternaire.

Une fois la boucle terminée, on imprime à l'écran la séquence des caractères du texte compressé.

```
> compresser:=proc(t) local s,calculerTriplet,i;
    global A,L,C;
    s:=NULL;

  calculerTriplet:=(a,b,c)->(poidsFort(a),poidsFaible(a),poidsFort
  (b),poidsFaible(b),c);
    i:=0; C:=t[9];
    while C <= 2 do
      motifMax(t,i,i+9,8);
      s:=(s,calculerTriplet(A,L,C));
      i:=i+L+1
    od;
    print(s)
  end;
```

Question 8

Le texte **tc** est divisé en blocs consécutifs de 5 caractères. Chaque bloc (indexé par l'indice **j** de sa première cellule dans **tc**) contient en substance :
- les 2 premiers caractères sont les chiffres en base 3 du décalage **k** à effectuer par rapport à la position courante **i** pour trouver dans le texte **s** en cours de reconstitution le début de la chaîne de longueur **l** à dupliquer à l'extrémité de **s.**
- les 2 caractères suivants sont les chiffres en base 3 de **l**.
- le dernier caractère est un caractère intermédiaire c, qui a été simplement copié lors de la compression, et qu'il suffit donc de placer en fin de **s** après duplication de la chaîne précédente. De plus, lorsque ce caractère est **x**, on peut interrompre la boucle.

A chaque tour de boucle, il convient d'actualiser la variable **i** en se décalant d'une longueur **l**, plus 1 pour tenir compte de la copie du caractère intermédiaire c.
Et bien sûr il faut actualiser **j** en le décalant de 5 unités pour passer au bloc suivant dans **tc**.
On notera qu'il est important de réactualiser **s** à chaque tour de la boucle indexée par **p**, et non pas seulement après lecture de chaque bloc, car certaines chaînes à dupliquer peuvent "se chevaucher" (voir l'exemple de compression développé dans l'énoncé).

```
> décompresser:=proc(tc) local s,i,j,k,l,p;
    s:=0,0,0,0,0,0,0,0,0; # texte décompressé en cours de
  formation
    i:=0; # position de début de la sous-chaîne
    j:=0; # n° d'ordre du bloc de 5 caractères ternaires en cours
  de lecture dans tc
    while true do # l'arrêt de la boucle se fera par l'instruction
  "break"
      k:=entier(tc[j],tc[j+1]); l:=entier(tc[j+2],tc[j+3]);
      for p from i+k+1 to i+k+l do s:=s,op(p,[s]) od;
      s:=s,tc[j+4];
      if tc[j+4] > 2 then break else i:=i+l+1; j:=j+5 fi
    od;
    print(s)
  end;
```