Épreuve d'informatique de l'X - 2011 - PT/PSI

Partie I. Opérations élémentaires

Question 1. [1] On commence par créer une image de même hauteur, largeur et profondeur que l'image de i (ligne 3). Puis, dans la boucle des lignes 4 à 8, on affecte le ton du pixel de l'image inversée.

```
1  inverser := proc(i)
2  local image, l, c;
3  image := allouer(i.H,i.L,i.P);
4  for l from 0 to i.H-1 do
5     for c from 0 to i.L-1 do
6        image.M[l,c]:= i.P - (i.M)[l,c]
7     od;
8  od;
9  return(image);
10 end;
```

Question 2. On commence par créer une image de même hauteur, largeur et profondeur que l'image de i (ligne 3). Puis, dans la boucle des lignes 4 à 8, on affecte le ton du pixel de l'image obtenue par symétrie d'axe vertical.

```
1  flipH := proc(i)
2  local image, c, l;
3  image := allouer(i.H,i.L,i.P);
4  for c from 0 to i.L-1 do
5     for l from 0 to i.H-1 do
6        image.M[l,c] := (i.M)[l, i.L-c-1]
7     od;
8  od;
9  return(image);
10 end;
```

1. les programmes ont été écrits dans un pseudo langage *Maple*, puisque l'accès des composantes de l'image ne peuvent se faire comme cela est décrit par l'énoncé. Si voulez voir un exécutable correct, consultez la solution que je propose en *Caml*.

Question 3 : On commence par créer une image de même largeur et profondeur que l'image de i et qui a pour hauteur la somme des hauteurs des images i_1 et i_2 (ligne 3). Puis, dans la boucle des lignes 4 à 11, on duplique les images i_1 et i_2 dans l'image résultat.

```
1  poserV := proc(i1,i2)
2  local image, c, l;
3  image := allouer(i1.H+i2.H,i1.L,i1.P);
4  for c from 0 to i1.L-1 do
5     for l from 0 to i1.H-1 do
6        image.M[l,c] := (i1.M)[l,c]
7     od;
8     for l from 0 to i2.H-1 do
9        image.M[l+i1.H,c] := (i2.M)[l,c]
10    od;
11 od;
12 return(image);
13 end;
```

Question 4 : On commence par créer une image de même hauteur et profondeur que l'image de i et qui a pour largeur la somme des largeurs des images i_1 et i_2 (ligne 3). Puis, dans la boucle des lignes 4 à 11, on duplique les images i_1 et i_2 dans l'image résultat.

```
1  poserH := proc(i1,i2)
2  local image, c, l;
3  image := allouer(i1.H,i1.L+i2.L,i1.P);
4  for l from 0 to i1.H-1 do
5     for c from 0 to i1.L-1 do
6        image.M[l,c] := (i1.M)[l,c]
7     od;
8     for c from 0 to i2.L-1 do
9        image.M[l,c+i1.L] := (i2.M)[l,c]
10    od;
11 od;
12 return(image);
13 end;
```

Partie II. Transferts

Question 5 : On commence par créer une image de même hauteur, largeur et profondeur que l'image de i (ligne 3). Puis, dans la boucle des lignes 4 à 8, on affecte le ton résultat de l'application de la fonction de transfert au ton du pixel de l'image de départ.

```
1  transferer := proc(i,P',t)
2  local image, c, l;
3  image := allouer(i.H,i.L,P');
4  for c from 0 to i.L-1 do
5     for l from 0 to i.H-1 do
6        image.M[l,c] := t[i.M[l,c]]
7     od;
8  od;
9  return(image);
10 end;
```

Question 6 : On commence par créer le tableau `t` correspondant à la fonction de transfert d'inversion d'une image de profondeur P (ligne 3 à 6). Il suffit ensuite appliquer la fonction de transfert à l'image i (ligne 7).

```
1  inverser := proc(i)
2  local t, k;
3  t := array(0..i.P);
4  for k from 0 to i.P do
5     t[k] := i.P-k
6  od;
7  return(transferer (i,i.P,t));
8  end;
```

Question 7 : Pour la fonction `histogramme`, on commence par initialiser le tableau `h` (boucle de la ligne 3). Ensuite, dans la boucle des lignes 4 à 9, on parcourt tous les pixels de l'image et on incrémente la case du tableau `h` correspondant au ton du pixel.

```
1  histogramme := proc(i,h)
2  local c, l, k, niveau;
3  for k from 0 to i.P do h[k] := 0 od;
4  for c from 0 to i.L-1 do
5     for l from 0 to i.H-1 do
6        niveau := (i.M)[l,c];
7        h[niveau] := h[niveau] + 1
8     od;
9  od;
10 return(h);
11 end;
```

Question 8 : On commence par récupérer l'histogramme h de l'image i (ligne 4). Puis à la ligne 6, on trouve `v_min` le premier indice k tel que $h[k]$ soit non nul (donc le v_{min} de l'énoncé). Dans la boucle des lignes 10 à 13, on calcule la fonction de transfert correspondant à l'égalisation. La variable s correspond à la somme $\sum_{k=0}^{k=v} h[k]$. Enfin, à la ligne 14, on applique la fonction de transfert pour calculer la nouvelle image qui est i dont tous les tons ont été égalisés.

```
1  egaliser := proc(i)
2  local h, v_min, v_prime, v, s, quotient;
3  h := array(0..i.P);
4  h := histogramme(i,h);
5  v_min := 0;
6  while h[v_min]=0 do v_min := v_min+1 od;
7  v_prime := array(0..P);
8  s := 0;
9  quotient := i.H*i.L - h[v_min];
10 for v from v_min to i.P do
11     s := s + h[v];
12     v_prime[v] := arrondir(i.P*(s - h[v_min])/quotient)
13 od;
14 return(transferer (i,i.P,v_prime));
15 end;
```

Question 9 : Dans le cas d'une image uniformément blanche, v_{min} est égal à la profondeur P et $\sum_{k=0}^{k=P} h[k] = h[P] = h[v_{min}] = H.L$. Il y a donc un problème de division par zéro quand on calcule l'égalisée d'une image uniformément blanche.

Question 10 : On crée la fonction de transfert correspondant à la réduction de profondeur. Pour chaque ton k, on cherche $v'[k]$ tel que $\left|\frac{v'[k]}{P'} - \frac{k}{P}\right|$ soit minimal, ce qui revient à minimiser $|v'[k]\,P - k\,P'|$ ou bien à trouver $v'[k]$, l'entier le plus proche de $\frac{k.P'}{P}$. Pour cela, on utilise la fonction `arrondi` à la ligne 5. Une fois la fonction de transfert calculée, il reste à appliquer la fonction `transferer` pour obtenir le résultat demandé (ligne 7).

```
1 reduire := proc(i,P_prime)
2 local v_prime, k;
3 v_prime := array(0..i.P);
4 for k from 0 to i.P do
5     v_prime[k] := arrondi(P_prime*k / i.P);
6 od;
7 transferer(i,P_prime,v_prime);
8 end;
```

Partie III. Tramage

Question 11 : On commence par créer une image de la même dimension que i et de profondeur 1 (ligne 3). On récupère la longueur du motif (ligne 4) et la largeur du motif (ligne 5). Dans la boucle des lignes 6 à 16, on remplit l'image de 0 ou de 1 suivant que le point correspondant dans l'image de départ i à un ton supérieur ou non au ton pixel du motif associé. Le seuil appliqué est choisi en fonction de la position du pixel traité dans l'image et du motif supposé dupliqué.

```
1  tramer := proc(i,m)
2  local image, longueur_motif, largeur_motif, l, c, seuil;
3  image := allouer(i.H,i.L,1);
4  longueur_motif := m.H;
5  largeur_motif := m.L;
6  for l from 0 to i.H-1 do
7     for c from 0 to i.L-1 do
8        seuil := m.M[l mod longueur_motif, c mod largeur_motif];
9        if (i.M)[l,c] > seuil
10           then
11              image.M[l,c] := 1
12           else
13              image.M[l,c] := 0;
14       fi;
15    od;
16 od;
17 return(image);
18 end;
```

Question 12 : Pour la fonction `tramerTelevision`, on crée le motif adapté à la profondeur de l'image à tramer (lignes 4 à 7). Ensuite, on applique la fonction `tramer` de la question précédente avec le motif calculé.

```
1 tramerTelevision := proc(i)
2 local n, motif, k;
3 n := i.P;
4 motif := allouer(n,1,n);
5 for k from 0 to n-1 do
6    motif.M[k,0] := k
7 od;
8 tramer(i,motif);
9 end;
```

Question 13 : La variable `deuxQuarts` étant supposé connu, on crée l'image 16×16 en accolant quatre fois l'image `deuxQuart` éventuellement symétrisée (lignes 3 à 5). Il suffit ensuite d'appliquer la fonction `tramer` avec le motif calculé à partir de l'image `deuxQuarts` (ligne 6).

```
1 tramerJournal := proc(i)
2 local img1, img2, motif;
3   img1 := poserV (deuxQuarts, flipH(deuxQuarts));
4   img2 := poserV (flipH(deuxQuarts), deuxQuarts);
5   motif := poseH(img1, img2);
6   tramer(i, motif);
7 end;
```

Question 14 : L'image N est une image de profondeur 1 comme l'image M mais la taille du motif a été réduite d'un facteur 2. Je n'ai pas compris ce qui est attendu par l'auteur de l'énoncé mais je vous propose une solution qui donne un aspect visuel ressemblant. On commence par calculer une image agrandie d'un facteur deux. On applique la fonction de tramage à l'image avec le motif diagonal. On prend soin de faire coïncider les profondeurs de l'image à traiter et du motif. Si l'image a une profondeur supérieure à celle de `MotifDiagonal`, on réduit l'image (ligne 17) et sinon on effectue un réduction sur le motif (ligne 18). Le résultat est une image de profondeur 1, deux fois plus grande que l'image de départ.

```
1  tramerJournal_bis :=  proc(i)
2  local l, c, img1, img2, MotifDiagonal, image, nuance;
3  img1 := poserV(deuxQuarts, flipH(deuxQuarts));
4  img2 := poserV(flipH(deuxQuarts), deuxQuarts);
5  MotifDiagonal := poserH(img1, img2);
6  image := allouer(2*i.H,2*i.L, i.P);
7  for l from 0 to i.H-1 do
8     for c from 0 to i.L-1 do
9              nuance := i.M.[l, c];
10             image.M[2*l, 2*c] := nuance;
11             image.M[2*l+1,2*c] := nuance;
12             image.M[2*l, 2*c+1] := nuance;
13             image.M[2*l+1,2*c+1] := nuance;
14    od;
15 od;
16   if i.P = 15 then tramer(image, MotifDiagonal)
17 elif i.P > 15 then tramer(reduire(image,15), MotifDiagonal)
18                  else tramer(image, reduire(MotifDiagonal, i.P))
19 fi;
20 end;
```