GU : Licence Sciences Technologies (semestre 5) **UE :** INF209
Épreuve de : Algorithmique Avancée
Date : 4 janvier 2007, 14h **Durée :** 3h
Documents autorisés
Épreuve de Monsieur Robson

1 [6 points]

Des arbres binaires peuvent être representés par des structures de noeuds, chaque noeud possédant un champ `valeur:entier` et deux champs `gauche` et `droit` qui seront des pointeurs vers les sous-arbres gauche et droit (ou `nul`).
Ecrire en exalgo

i. une fonction booléenne avec deux paramètres qui seront des pointeurs vers les racines de deux arbres et qui retourne vrai si les deux arbres sont identiques (même forme et mêmes valeurs).

```
fonction identique(val P,Q: arbreptr):booleen
debut
selon que
   cas P=nul ou Q=nul: retourne P=Q
   cas P^.valeur!=Q^.valeur: retourne faux
   cas !identique(P^.gauche,Q^.gauche): retourne faux
   autrement retourne identique(P^.droit,Q^.droit)
finselonque
finfonction
```

ii. une fonction avec un paramètre, un pointeur vers la racine d'un arbre, et qui construit une copie identique de l'arbre pointé et retourne un pointeur vers la racine de cette copie.

```
fonction copie(val P: arbreptr): arbreptr
var cop: arbreptr;
debut
si P=nul alors retourne nul finsi;
allouer(cop);
cop^.valeur:=P^.valeur;
cop^.gauche:=copie(P^.gauche);
cop^.droit:=copie(P^.droit);
retourne cop
finfonction
```

2 [5 points]

Une procédure récursive aléatoire `P(l1,w1,l2,w2)` se comporte comme :

```
si l1=0 et l2=0 alors retourne finsi;
si l1=0 alors w1:=0 finsi;
si l2=0 alors w2:=0 finsi;
Pause pour une durée aléatoire avec espérance (moyenne) de 1/(w1+w2);
avec probabilité w1/(w1+w2) faire P(l1-1,w1+1,l2,w2)
autrement (donc avec probabilité w2/(w1+w2)) faire P(l1,w1,l2-1,w2+1);
retourne;
```

On veut calculer la somme en moyenne des durées de toutes les pauses quand `P(100,1,100,1)` est appelé.

i. Donner une récurrence pour cette somme moyenne dans le cas général d'un appel `P(x1,y1,x2,y2)`
$T(0, y1, 0, y2) = 0$
$T(x1, y1, 0, y2) = 1/y1 + T(x1 - 1, y1 + 1, 0, 0)$
$T(0, y1, x2, y2) = 1/y2 + T(0, 0, x2 - 1, y2 + 1)$
$T(x1, y1, x2, y2) = 1/(y1 + y2) + (y1 * T(x1 - 1, y1 + 1, x2, y2) + y2 * T(x1, y1, x2 - 1, y2 + 1))/(y1 + y2)$

ii. Donner une procédure pour calculer la somme demandée en utilisant la programmation dynamique.

```
procedure CALC()
var T:tab; /* un tableau suffisamment grand */
debut
pour y1 allant de 0 a 101 par pas de 1 faire
   pour y2 allant de 0 a 101 par pas de 1 faire
      T[0][y1][0][y2]:=0
   finpour
finpour;
pour x allant de 0 a 100 par pas de 1 faire
   pour y1 allant de 0 a 101 par pas de 1 faire
      pour y2 allant de 0 a 101 par pas de 1 faire
         T[x][y1][0][y2]:=1/y1 + T[x-1][y1+1][0][0]
         T[0][y1][x][y2]:=1/y2 + T[0][0][x-1][y2+1]
      finpour
   finpour
finpour;
pour x1 allant de 1 a 100 par pas de 1 faire
   pour x2 allant de 1 a 100 par pas de 1 faire
      pour y1 allant de 1 a 101 par pas de 1 faire
         pour y2 allant de 1 a 101 par pas de 1 faire
            T[x1][y1][x2][y2]:=(1+y1*T[x1-1][y1+1][x2][y2]+y2*T[x1][y1][x2-1][y2+1])/(y1+y2)
         finpour
      finpour
   finpour
finpour;
/* la somme souhaitée est T[100][1][100][1] */
finprocedure
```

3 [4 points]

i. Donner des affectations d'initialisation et une boucle `tant que` qui calculent le plus petit entier i tel que la somme $1 + \cdots + i$ soit supérieure à un entier positif donné m.

```
i:=1;
somme:=1;
tant que somme <= m faire
   i:=i+1;
   somme:=somme+i
fin tantque;
```

ii. Donner un invariant pour votre boucle et expliquer (sans démonstration) comment cet invariant peut être utilisé pour prouver que la boucle soit correcte.
$i > 0$ et $somme = 1 + \cdots + i$ et $somme <= m + i$
- L'invariant est une conséquence de la précondition $i = 1$ *et* $somme = 1$
- Si l'invariant et la condition de la boucle ($somme <= m$) sont vrais avant une exécution de `i:=i+1; somme:=somme+i` l'invariant reste vraie après

– La postcondition (i est le plus petit ...) est une conséquence de l'invariant plus la négation de la condition de la boucle ($somme > m$)

4 5 [points]

Déterminer (en fonction de n à $O(\ )$ près) le temps de calcul des boucles (a), (b), (c) (d), et (e)
(`P()` est une procédure dont le temps de calcul est constant.)

i.
```
pour i allant de 1 a n*n*n par pas de 1 faire P(); finpour;
```
$O(n^3)$

ii.
```
pour i allant de 1 a n par pas de 1 faire
   pour j allant de 1 a i par pas de 2 faire P(); finpour;
finpour;
```
$O(n^2)$

iii.
```
i:=1;
tant que i*i<n faire i:=i*2;
fintantque;
```
$O(\log n)$

iv.
```
pour i allant de 1 a n par pas de 1 faire
   j:=1;
   tant que i*j<n faire j:=j+2;
   fintantque;
finpour;
```
$O(n \log n)$ $\qquad (\approx (n + n/2 + n/3 + ... + n/n)/2)$

v.
```
m:=1;
pour k allant de 1 a n par pas de 1 faire
   i:=1;
   tant que i<k faire
      i:=i*2;
      m:=m*2;
   fintantque;
   pour j allant de 1 a m par pas de 1 faire P(); finpour;
finpour;
```
Difficile. Le temps est dominé par la boucle `pour` quand $k = n$ et, donc, le temps est $O(derniere\ valeur\ de\ m)$. Cette dernière valeur de m est le produit de n nombres qui sont les k arrondi vers le haut à la prochaine puissance de 2.
$O((2n)^n)$ est une assez bonne réponse (vaut le point) ;
$O(n!\ 2^n)$ est meilleure mais pas la meilleure possible.