GU : Licence Sciences Technologies (semestre 5) **UE :** INF209
Épreuve de : Algorithmique Avancée
Date : 11 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`).
On peut définir un ordre sur les arbres binaires tel que l'arbre vide précède tout autre arbre et, entre deux arbres non vides, l'un précède l'autre si sa valeur est inférieur, ou (dans le cas de deux valeurs égales) son sous-arbre gauche précède celui de l'autre, ou (dans les cas de deux sous-arbres gauches égaux) son sous-arbre droit précède celui de l'autre.
Ecrire en exalgo

i. une fonction entière avec deux paramètres qui seront des pointeurs vers les racines de deux arbres (éventuellement vides) et qui retourne -1 si le premier arbre précède l'autre dans cet ordre, 0 si les deux sont identiques et $+1$ dans le dernier cas où le deuxième précède le premier.

ii. une procédure avec un paramètre, un tableaux de pointeurs vers la racine d'un arbre, et qui trie les pointeurs selon l'ordre de leur arbre, en utilisant la fonction de la partie précédente. (On n'exige pas que l'algorithme de tri soit efficace.)

2 [5 points]

On veut calculer la probabilité qu'un arbre binaire de recherche aléatoire de 1000 noeuds ait son hauteur strictement inférieure à 50.

i. Donner une récurrence pour cette probabilité dans le cas général d'un arbre de n noeuds et hauteur inférieure à h.
Vous devrez utiliser le fait que le sous-arbre gauche de cet arbre a un nombre g de noeuds avec probabilité $1/n$ pour tout g entre 0 et $n-1$.

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

3 [4 points]

i. Donner des affectations d'initialisation et une boucle `tant que` qui calculent le plus grand entier i tel que $i! + (2i)!$ soit inférieure à un entier positif donné m.

ii. Donner un invariant pour votre boucle et expliquer (sans démonstration) comment cet invariant peut être utilisé pour prouver que la boucle soit correcte.

4 5 [points]

Déterminer (en fonction de n à $O(\ )$ près) le temps de calcul des boucles (i), (ii), (iii), (iv) et (v)
(`P()` est une procédure dont le temps de calcul est constant.)

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

ii.
```
pour i allant de 1 a n par pas de 1 faire
   pour j allant de 1 a i par pas de 1 faire
      pour k allant de j a i par pas de 1 faire P()
      finpour
   finpour;
finpour;
```

iii.
```
i:=n;
tant que i>1 faire
   si i mod 2 = 0 alors
      i:=i/2
   sinon
      i:=i-1
   finsi
fintantque;
```

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;
```

v.
```
i:=1;
tant que i<n*n faire
   i:=i+1+i/n /* arrondi vers le bas */
fintantque
```