Année 2004-2005 — Session de Juin 2005

GU : Licence Sciences Technologies (semestre 5) — **UE :** INF209
Épreuve de : Algorithmique Avancée
Date : 15 juin 2005 — **Durée :** 3h
Documents autorisés
Épreuve de Monsieur Robson

1 [5 points]

Je veux calculer A^n, c'est-à-dire la n-ème puissance d'une matrice (de taille $m \times m$) A ; je dispose d'une procédure $Matmult(X, Y, Z)$ qui calcule en Z le produit des deux matrices X et Y. Au lieu de faire $n-1$ multiplications, je veux utiliser le fait que A^n pour $n > 1$ est soit le produit de $A^{n/2}$ avec lui-même (quand n est pair), soit le produit de A^{n-1} avec A (à utiliser quand n est impair.)

Ecrire une procédure récursive $Puissance(A, n, B)$ qui effectue ce calcul.

2 [4 points]

Je peux calculer le nombre de formes possibles d'un arbre binaire de n sommets, en utilisant la récurrence $Arb(n) = \sum_{g=0}^{n-1} Arb(g) * Arb(n-1-g)$ (où g est la taille du sous-arbre gauche et $n-1-g$ celle de celui de droite) ; avec la condition $Arb(0) = 1$.

Expliquer comment la programmation dynamique permet de faire ce calcul de façon beaucoup plus efficace qu'une fonction récursive appliquant directement la récurrence.

3 [4 points]

le code qui suit est censé déterminer si le nombre positif n est premier (c'est-à-dire qu'il n'existe pas de facteur f de n tel que $2 \le f < n$.)

Proposer un invariant qui pourrait être utilisé pour démontrer que la boucle est correcte.

```
f:=2;
tant que f < n et (n/f)*f < n faire // f n'est pas facteur de n
   f:=f+1
fintantque;
// n est premier si (et seulement si) f=n
```

4 [3 points]

Donner le temps de calcul des boucles qui suivent en termes de $O(fonction\ de\ n)$:

1.
```
i:=1;
tant que i<n faire
   i:=i*10
fintantque;
```
2.
```
pour i allant de 1 à n faire
   pour j allant de i à n faire
      pour k allant de i à j faire
         x:=x+1;
      finpour
   finpour
finpour;
```

3. ```
   j:=1;
   pour 1 allant de 1 à n faire
      tant que j<i faire
         j:=j*2
      fintantque
   finpour;
   ```

5 [4 points]

le type listep est un pointeur vers une structure qui contient un entier (*premier*) et un pointeur du même type vers le prochain elément (*reste*) ; la liste vide est représentée par le pointeur $NULL$. On suppose que deux variables de ce type A et B pointent, toutes les deux vers des listes où les entiers sont triés en ordre strictement croissant. Ecrire des instructions pour décider si c'est le cas que tous les entiers de A sont aussi présents en B. (Une solution correcte et efficace est souhaitée.)
   ```