INF4100 — Analyse et conception d'algorithmes
Examen intra (Hiver 2008)

Durée: (3 heures) **Documentation autorisée:** Toute documentation personnelle.

Nom: _____

Code permanent: | | | | | | | | | | | | |

Directives:

a. Répondez aux questions directement sur le questionnaire dans les espaces prévus à cette fin.

b. Utilisez les versos pour vos brouillons. Si vous devez utiliser un verso pour compléter une réponse, indiquez le au recto.

c. N'écrivez rien dans la table à droite (réservée au correcteur).

1	/10
2	/10
3	/10
4	/15
5	/10
Bonus	/5
Total	/55

1. Définition de O (10 pts)

Pour les fonctions f_1 et f_2 indiquées plus bas, donnez un estimé O *le plus précis et le plus simple possible*. *Justifiez* votre réponse en référant de façon détaillée à la définition de O et à la fonction f_i — en d'autres mots, identifiez les constantes N et c telles que la définition de O s'applique pour la fonction f_i.

$$f_1(n) = n^2 \cdot 3^{n+2} + 2^n + 10^5$$

$$f_2(n) \quad = \quad \lg\,(n^{2n} \cdot 2^{n+1}) + n^2$$

2. Analyse d'un algorithme itératif (10 pts)

Soit l'algorithme suivant (n'essayez pas de le comprendre ;) :

```
procedure f( int a[*], int n ) returns int m
{
   m = high(int);
   for [i = 1 to n] {
     for [j = 1 to i] {
       int r = 0;
       for [k = 1 to n] {
         r = r + a[k];
       }
       m = min( m, r );
     }
   }
}
```

Identifiez une *opération barométrique* appropriée pour cet algorithme, déterminez le nombre exact de fois où cette opération barométrique est exécutée, puis trouvez un estimé Θ.

Note : Vous devez indiquer *de façon explicite* la sommation qui représente le nombre exact d'opérations barométriques. Vous devez aussi indiquer les étapes qui permettent d'arriver au résultat (exact) final. Quant au résultat asymptotique, seule la classe de complexité suffit (pas besoin de recourir à la définition de O).

3. Analyse d'un algorithme récursif (10 pts)

Soit `C` une *constante* entière *supérieure ou égale à 1.*
Soit l'algorithme récursif présenté plus bas. Pour faire son analyse, on choisit comme opérations barométriques les *utilisations des opérateurs* `min` *et* `max`. On suppose aussi que la procédure auxiliaire `traiterTableau` — dont le code est omis — exécute *exactement* $\frac{n}{2C}$ opérations barométriques.

```
procedure f( ref int a[*], int inf, int sup ) returns int r
{
  if ( sup - inf + 1 <= 10 ) {
    r = high(int);
    for [k = inf to sup] {
      r = min( r, a[k] );
    }
  } else {
    int m = (sup - inf + 1) / (5*C);
    r = low(int);
    int bInf = inf;
    for [k = 1 to 5] {
      int r0 = f( a, bInf, bInf + m - 1 );
      r  = max( r, r0 );
      bInf += m;
    }
    traiterTableau( a, inf, sup )
  }
}
```

a. Donnez les équations de récurrence décrivant le temps d'exécution de la fonction `f`. Spécifiez clairement les différents cas pour n (cas de base vs. cas récursifs).

b. Donnez la solution asymptotique Θ pour le temps d'exécution de `f` en fonction de `C` en vous basant sur les équations de récurrence spécifiées en a. Pour ce faire, utilisez le théorème général, en décrivant brièvement comment il s'applique (valeur de a, b et k et leurs relations), et ce en fonction des différentes valeurs possibles pour `C` (entier supérieur ou égal à 1).

4. Création et analyse d'un algorithme récursif (15 pts)

On veut écrire une procédure `minMax` qui reçoit en arguments un tableau `a` d'entier et un entier `n` (taille du tableau), et qui retourne deux résultats :

- `mi` = La valeur minimum parmi les `n` éléments de `a`.
- `ma` = La valeur maximum parmi les `n` éléments de `a`.

```
procedure minMax( int a[*], int n, res int mi, res int ma )
# PRECONDITION
#   n = nombre d'elements de a
# POSTCONDITION
#   mi = MINIMUM( 1 <= i <= n :: a[i] )
#   ma = MAXIMUM( 1 <= i <= n :: a[i] )
```

a. Donnez, en MPD, une mise en oeuvre *récursive dichotomique équilibrée* (décomposition en deux sous-problèmes de tailles équivalentes) pour la procédure `minMax`. Vous pouvez évidemment introduire une ou des procédures auxiliaires. (Écrivez votre algorithme sur la page suivante.)

   Bonus : Un bonus (max. 5 pts) sera accordé pour une procédure qui utilise un *seuil* (spécifié par `TAILLE_SIMPLE`) pour déterminer le point à partir duquel la récursion se termine et une solution itérative est utilisée. L'analyse asymptotique devra aussi être faite en conséquence.

b. Donnez les équations de récurrence associées à l'analyse asymptotique de votre procédure, en indiquant explicitement la ou les opérations barométriques.

c. Donnez la solution asymptotique à vos équations de récurrence.

```
procedure minMax( int a[*], int n, res int mi, res int ma )

 # PRECONDITION
 #   n = nombre d'elements de a
 # POSTCONDITION
 #   mi = MINIMUM( 1 <= i <= n :: a[i] )
 #   ma = MAXIMUM( 1 <= i <= n :: a[i] )
```

5. Analyse d'un algorithme récursif (10 pts)

Soit la procédure récursive `f` suivante :

```
procedure f( ref int a[*], int inf, int sup )
{
  if ( inf <= sup ) {
    a[inf] :=: a[sup];
    f( a, inf+1, sup-1 );
  }
}
```

a. Que fait cette procédure?

b. Donnez les équations de récurrence associées à l'analyse asymptotique de cette procédure, en indiquant explicitement la ou les opérations barométriques appropriées.

c. Quelle est la solution à vos équations de récurrence? Justifiez votre réponse en utilisant la méthode de résolution par substitution. Indiquez aussi la condition appropriée sur `n`.