Corrigé Examen Final Master MIL 2009-2010

Exercice1 :

a) 1ère solution :

```
liste somme(liste p, liste q)
{ liste L=NULL;
  if(p==NULL && q==NULL) return(NULL);
  if (p==NULL) return(q);
  else if(q==NULL)  return(p);
        else { if(p->exp>q->exp)
                      { L=somme(p->svt,q); ajoutermonome(&L,p->exp,q->coef);}
                 else if (p->exp<q->exp)
                           { L=somme(p,q->svt); ajoutermonome(&L,q->exp,q->coef);}
                      else {L=somme(p->svt,q->svt);
                            ajoutermonome(&L,q->exp,p->coef+q->coef);}
                  return(L);
               }
}
```

2ème solution :

```
void somme(liste p, liste q, liste *L)     /* 1er appel  L=NULL  */
{  if (p==NULL) *L=q;
  else if(q==NULL)  *L=p;
       else if(p->exp>q->exp)
                    { somme(p->svt,q,L);inseretete(L,p->exp,q->coef); }
              else if (p->exp<q->exp)
                           { somme(p,q->svt,L); inseretete(L,q->exp,q->coef);}
                   else {somme(p->svt,q->svt,L);inseretete(L,q->exp,p->coef+q->coef); }
}
```

b) <u>Complexité :</u>

$$\text{Somme}(p,q) = \begin{cases} c \quad \text{si} \quad p=\text{NULL} \\ c \quad \text{si} \quad q=\text{NULL} \\ c+\text{somme}(p\text{->svt},q) \quad \text{si} \quad p\text{->exp} > q\text{->exp} \\ c+\text{somme}(p,q\text{->svt}) \quad \text{si} \quad p\text{->exp}<q\text{->exp} \\ c+\text{somme}(p\text{->svt},q\text{->svt}) \quad \text{si } p\text{->exp}=q\text{->exp} \end{cases}$$

Au pire cas *<u>tous les exposants sont différents</u>*.
Si le nombre d'éléments de p est n et le nombre d'éléments de q est m donc au pire cas on va ajouter n+m monômes dans la liste somme.
Donc la complexité= O(n+m) ;

c)
```
int puissance(int x0, int n)
{   int p=1 ;
     for( ; n>0 ; n--)  p=p*x0 ;
     return(p) ;
}
int   Evaluation( liste p, int x0)
{  int s=0 ;
    while(p !=NULL)
       { s=s+p->coef*puissance(x0,p->exp) ;
         P=p->svt ;
       }
    return(s) ;
}
```

d) <u>Complexité :</u>
Etant donnée que la fonction puissance est en $\Theta(n)$ et si le nombre d'éléments de p est n donc la fonction puissance s'exécute n fois donc la complexité de la fonction Evaluation est en $\Theta(n^2)$.

e)
```
int EvalHorner( liste p, int x0)
{  int s=0 ;
   while(p !=NULL)
   {  s=s*x0+ p->coef ;
       P=p->svt ;
   }
return(s) ;
}
```

f) <u>Complexité :</u>
Si le nombre d'éléments de p est n et comme il n'y a plus de fonction puissance donc la fonction EvalHorner effectue n sommes donc la complexité est en $\Theta(n)$.

Exercice2 :

a) On montre par récurrence que $\sum_{i=0}^{n-1} 2^i = 2^n - 1$

Pour n-1=0 on a :

$\sum_{i=0}^{0} 2^i = 2^0 = 1$ et $2^n - 1 = 2^1 - 1 = 1$ donc l'équation est vraie pour n-1=0

On suppose qu'elle est vraie pour n-1 et on montre qu'elle est vraie pour n

On montre que $\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$

$\sum_{i=0}^{n} 2^i = \sum_{i=0}^{n-1} 2^i + 2^n$

$= 2^n - 1 + 2^n$

$= 2*2^n - 1$

$= 2^1 * 2^n - 1$

$= 2^{n+1} - 1$ donc l'équation $\sum_{i=0}^{n-1} 2^i = 2^n - 1$ est vraie

b) $T(n)=T(n-1) + 2^n$

$T(0)=1$

$T(n)=T(n-1)+2^n$

$=T(n-2)+2^{n-1}+2^n$

$= T(n-3)+2^{n-2}+2^{n-1}+2^n$

$= \dots$

$= T(0) + 2^1 + 2^2 + \dots + 2^{n-2}+2^{n-1}+2^n$

$= 1 + \sum_{i=1}^{n} 2^i$

$= \sum_{i=0}^{n} 2^i$

$\mathbf{= 2^{n+1} - 1}$

c) $2^{f(n)} = \Theta(2^{g(n)})$?

On constate que si $f(n) = 2n = \Theta(n)$

Donc es-ce-que $2^{2n} = \Theta(2^n)$?

$2^{2n} = 2^n * 2^n$ ne peut pas être égal à $\Theta(2^n)$

Donc $2^{f(n)} \neq \Theta(2^{g(n)})$