Algo3 **DT1 : Complexity**

Exercise 01: Let the real matrices n x n be : A=(aij), B=(bij) and C=(cij). Study the complexity of the following algorithm which calculates the coefficients (cij) of the product matrix C= A x B according to the classical formula: $c_{ij} = \sum_{k=1}^{n} a_{ik}.b_{kj}$ for i, j between 1 and n

```
const int n=8;
typedef float matrice[n][n];
void multimat(matrice a, matrice b, matrice c)
{for (int i=0; i<n; i++)
                    for (int j=0; j<n; j++)
                             { c[i][j]=0;
                                       for (int k=0; k<n; k++)
            c[i][j]=c[i][j]+a[i][k]*b[k][j];
                             }
}
```

Exercise 02: Let the algorithm that calculates the value of the polynomial P(x, n)

$P(x, n) = \sum_{i=0}^{n} a_i x^i$, at a given point x.

We have 3 versions of this algorithm.

a)
```
p=a[0];
for (i=1; i<=n;i++){   xpi=puissance(x, i);
                       p=p+a[i]*xpi;}
```

b)
```
p=a[0]; xpi=1;
for (i=1; i<=n;i++){   xpi=xpi*x;
                       p=p+a[i]*xpi;}
```

c) Horner's Méthod
```
p=a[n];
for (i=n-1; i>=0;i--)     {     p=p*x+a[i]; }
```

Calculate the complexity of each version.

Exercice 03: Calculate the complexity of the following 2 algorithms for the Fibonacci sequence.

a)
```
int Fib (int n)
{ int tab[n+1];
tab[0]=1;          tab[1]=1;
for (int i=2; i<=n; i++) tab[i]=tab[i-1]+tab[i-2];
return tab[n];}
```

b)
```
int Fib (int n)
{ int tab[2];
tab[0]=1;          tab[1]=1;
for (int i=1; i<=n/2; i++) {tab[0]=tab[0]+tab[1];
                            tab[1]=tab[0]+tab[1];
if (pair(n)) return tab[0]; else return tab[1];}
```