Ex 2 TD 1

seq1: O(n)
seq2:O(n^2)
seq3:O(log(n))
seq4:O(nlog(n))
seq5:O($2^{log(n)}$)
seq6:O(2^n)
seq7:O(2^n)

Ex 6 TD 2

La condition d'arrêt est : if (x!=0)
a) Entrée : 5,7, 13,4,7,0,5,15,50,0
Affichage : ,0,7,4,13,7,5

Ex 7 TD 2

a)
void Binaire (int n)
{if (n!=0)
{ Binaire(n/2); cout <<n%2; }
}
B(13)-->B(6) et empile 1 -->B(3) et empile 0-->B(1) et empile 1-->B(0) et empile 1

1
1
0
1

On dépile et on affiche, d'ou l'affichage 1101 pour 13.

RQ:

Pour B(0), l'algorithme n'affiche rien.
Pour afficher B(0), il suffit d'éviter les accolades.
void Binaire (int n)
{if (n!=0)
Binaire(n/2);
cout <<n%2;
}

b) Binaire fait 3 opérations (en jaune) d'ou l'équation : B(n)=3+B(n/2)

Ex 8 TD2

a)
```
int palindrome (char *m, int g, int d)
{
if (m[g]!=m[d]) return 0;
else
if (g>=d) return 1;
return palindrome (m, g+1, d-1)
}
```

b)
T(n)=2+T(n-2)

Ex 5 TD 2
```
int somme (int X[], int n)
{
if (n==0) return 0;
return (X[n-1] + somme (X, n-1))
```
soit le tableau suivant de taille 4:

5	6	7	8

somme(X, 4)--> 8+somme(X,3)-->8+7+somme(2)-->8+7+6+somme(X,1)
-->8+7+6+5+somme(X,0) -->8+7+6+5+0=26
Equation récurrente de complexité en terme d'opération arithmétiques : T(n)=3+T(n-1)

Ex 3 TD 3
La courbe au début est croissante, ensuite décroissante. On fait un échantillonnage et on stocke les valeurs dans un tableau.

```
int max (int T[], int g, int d)
{
int m;
m=(g+d)/2;
if (T[n]>T[n-1] && T[n]>T[n+1])  return n;
if (T[n]>T[n-1]) return max(T, n+1, d);
return max(T, g, n-1)
}
```

T(n)=3+T(n/2)

Version récursive de Factoriel
```
Factoriel (int n)
{
if (n==0) return 1;
else
return n*Factoriel(n-1);
}
```
Il s'agit d'une récursivité non terminale, **pour la version dérécursivée, on utilise une pile** :

```
int Factoriel(int n)
{
p.init();
while(n!=0)
{
p.push(n);
n=n-1;    //traitement de la terminaison
}
res=1;
while (!p.empty())
{
x=p.pop();
res=res*x;
}
return res;
}
```

Version dérécursivée de la somme des éléments d'un tableau (Ex5 TD2)

```
int somme (int X[], int n)
{
while (n!=0)
{
p.push(n);
n=n-1;
}
res=0;
while (!p.empty())
{
u=p.pop();
res=res+X[u-1];
}
return res;
}
```