DEVOIR SURVEILLE

Module: RO-Complexité	**Niveau:** 4 Arctic-4Sim-4GL
Documents: Non autorisés	**Durée:** 1h00
Enseignants : R. Frefita, I. Denden, R. Guetari et S. Mesfar	**Date :** 29 Octobre 2014

Exercice-1 : (6 points)

Considérons la fonction **traiterChaine** présentée ci-dessous

```
void traiterChaine (char* str) {//str est une chaine de caractères
     int i = 0 ;
     int j ;
     int len = strlen (str) ;//strlen retourne la longueur de la chaine str

     while (i < len – 1) {
           if (str [i] == '/' && str [i + 1] == '/') {
              for (j = i + 1, j < len – 2 ; j++)
                  str [j] = str [j + 1] ;
              str [len – 1] = ' ';
           } else
                ++i ;
     }
}
```

1. Expliquer ce que fait la fonction **traiterChaine**().
2. Soit **str** une chaine de longueur ***n,*** donner le contenu de str qui maximise le nombre d'opérations exécutées par ce programme.
3. En déduire la complexité de ce programme au pire des cas à un *O* près.

Exercice-2: (6 points)

Considérons les programmes illustrés ci-dessous :

```
void P1(int n) {
int s = 0;

WHILE (n > 0){
     n = n/3;
     s = s + 1 ;
}
```

```
void P2(int n) {
int s = 0;

FOR(i = 0;i< n; i++) {
    FOR(j= i;j<= n*n;j++){
        s = s + 1;
    }
}
```

```
void P3(int n) {
int s = 0;

FOR(i = 0;i< n;i++) {
    FOR(j= 0;j< i*i; j++){
        FOR(k= 0;k<j;k++){
            s = s + 1;
        }
    }
}
```

4. Donner la complexité à un *O* près des programmes P1, P2 et P3 présentées ci-dessus.

Exercice-3: (8 points)

Considérons deux réels **a** et **b** tels que **a < b**. Etant donné une fonction ***f()*** dont les valeurs ***f***(a),***f***(a+1),***f***(a+2),...,***f***(b) sont connues. Une intégrale approchée de ***f()*** entre deux entiers a et b est égale à la somme des valeurs *f(a),f(a+1),...,f*(b).

1. Ecrire une fonction <u>récursive</u> « **float Integf (float tab[], int n)** ». Cette fonction prend comme paramètres le nombre de valeurs ***n*** de ***f()*** ainsi que l'ensemble des valeurs de ***f()*** stockées dans un tableau nommé **tab[]**. La fonction retourne comme résultat l'intégrale de **f()**.
2. Donner le type de récursivité de cet algorithme (terminale ou non). Justifier
3. Donner l'équation récurrente de la complexité en termes de nombre d'opérations pour la fonction **Integf()** (avec explication).
4. Ecrire une fonction itérative **float iterIntegf (float tab[],int n)** issue de la <u>dérécursivation</u> de la fonction **Integf()**.