Ecole préparatoire en Sciences et Techniques d'Oran	
Intitulé : Algorithmique 2 et structures de données avancées	Sigle : Algo2
2^{me} année, Semestre 3, 2011/2012	Examen de synthèse
Février 2012	Durée : 02h00mn

Partie cours :10 pts

Exercice 1 :4.5 pts

1. Soit la fonction suivante :

```
int f(int n, int p)
{
  int a1,a2;
  if ((p==0)||(p==n)) return 1;
  else
        {
          a1=f(n-1,p-1);
          a2=f(n-1,p);
          return a1+a2;
        }
}
```

   (a) donnez l'arbre d'exécution pour f(4,2).

   (b) si on met *a1* et *a2* comme variables globales, est ce qu'on obtient le même résultat (OUI/NON).

2. donnez un exemple de fonction terminale.

Exercice 2 : 5.5 pts

Considérons deux algorithmes A_1 et A_2 pour un même problème de taille N, avec $C_{A_1}(N) = 2N$ et $C_{A_2}(N) = 0.5N^2$:

1. tracez les courbes correspondant à C_{A_1} et C_{A_2},
2. que peut-on dire sur ces deux complexités ?
3. donnez la relation d'ordre existant entre C_{A_1} et C_{A_2} ? justifiez.

Partie TD : 10pts

Exercice 1 : 3.5 pts

Ecrire une fonction *occurence* qui a en paramètre une chaîne de caractères et qui renvoie par un return le nombre d'occurrences du caractère espace " ". Cette fonction devra parcourir la chaîne en utilisant un pointeur. Tester cette fonction en écrivant un programme qui calcule le nombre d'occurences de " ", dans un nombre quelconque de chaînes de caractères.

Exercice 2 : 3.75 pts

En utilisant les pointeurs, écrire un programme en C++ qui permet de :

1. saisir la taille N d'un tableau T dont les éléments sont de type *double*,
2. allouer l'espace mémoire suffisant pour le tableau T,
3. saisir les éléments du tableau T,
4. mettre dans un pointeur p l'adresse de la 5^{me} case du tableau,
5. afficher le contenu de ce tableau à partir de la 5^{me} case.

Exercice 3 : 1.75 pts

Soit le programme suivant suivante :

```
#include <iostream>
main(){
        char * jour[7] = {"lundi", "mardi", "mercredi", "jeudi", "vendredi", "samedi", "dimanche"};
        int i;
        cout << "Donnez un entier entre 1 et 7 :" ; cin >> i;
cout << "Le jour numéro " << i << "est : " << jour[i-1] << endl;
      }
```

Ajouter à ce programme les instructions nécessaires, pour :

1. que l'ordre des jours soit : "Samedi", "dimanche", "lundi", "mardi", "mercredi", "jeudi", "vendredi",
2. afficher tous les jours de 1 à 7.

Exercice 4 : 1.0 pt

Soit la fonction de Morris :

```
int morris (int m, int n){
     if (m == 0) return 1;
     else return morris(m - 1, morris(m, n));
}
```

Expliquer pourquoi l'appel de *morris(1,0)* ne se termine pas !

Ecole préparatoire en Sciences et Techniques d'Oran
Intitulé : Algorithmique 2 et structures de données avancées — Sigle : Algo2
2^{me} année, Semestre 3, 2011/2012 — Corrigé type de l'examen de synthèse
Février 2012 — Durée : 02h00mn

Partie cours : 8.5 pts

Exercice 1 :4.25 pts

1. L'arbre d'exécution : (0.25*10)

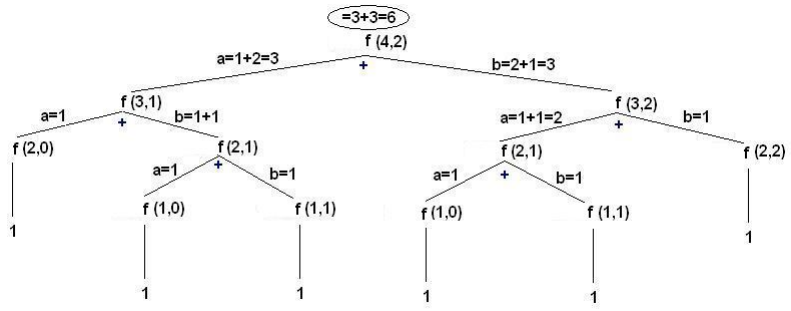

FIGURE 1 – L'arbre d'exécution de $f(4,2)$

2. NON. (0.25)
3. Un exemple de fonction récursive terminale : (0.25*6)

```
void enumerer(int x){
      if (n<=0) cout << "fin";
      else {
               cout << n;;
              enumerer(n-1);
             }
      }
```

Exercice 2 : 4.25 pts

1. les courbes correspondant à C_{A_1} et C_{A_2} : (0.75*2pts)

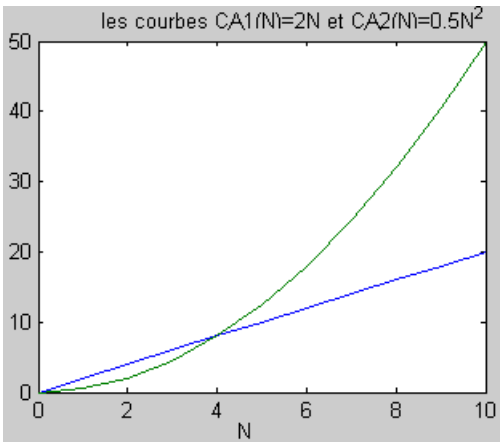

FIGURE 2 – Les courbes $C_{A_1}(N) = 2N$ et $C_{A_2}(N) = 0.5N^2$

2. l'analyse des deux complexités : (0.5*2+0.25)
   (a) A1 fait plus d'opérations que A2 pour $N < 4$,
   (b) dès que $N \geq 4$ A1 fait moins d'opérations que A2.

   donc on peut dire que l'algorithme A1 est meilleur que l'algorithme A2.
3. la relation d'ordre entre C_{A_1} et C_{A_2} : (0.5+1)
   $2N = O(0.5N^2)$ puisque : $\lim_{N\to+\infty} \frac{2N}{0.5N^2} = \lim_{N\to+\infty} \frac{4}{N} = 0$

Partie TD : 11.5pts

Exercice 1 : 4.25 pts

(2.0pts+2.25pt)

```
#include<iostream>
using namespace std;

int nba(char *p) (0.5pt)
{
    int n=0; (0.25)
    while (*p != '\0') (0.25pt)
    {
        if (*p ==' ') n++; (0.5pt)
        p++;(0.25pt)
    }
    return n;(0.25pt)
}

main() (0.25*9)
{
    char ss[50];
    int n;
    cout << "Donnez le nombre de chaînes de caractères : " ;
    cin << n;
    for(int i=0; i<n; i++){
cout << "Saisissez une chaine de caractères : " << endl;
    cin.getline(ss, 50);
    cout << "Vous avez saisi : " << ss << endl;
    cout << "La chaine comporte " << nba(ss) << " " << endl;
   }
}
```

Exercice 2 : 4.0 pts

Le programme en C++ :

```
main(){
    int N;   (0.25pt)
    cout << "N=" ;
    cin >> N; (0.25pt)
    double * T=new double[N]; (0.75pt)
    double *pt = T;
    for(int i=0; i<N; i++) (0.5pt)
        {
           cout << "T[" << i << "]=" ; cin >> *pt; (0.25pt)
           pt++; (0.25)
```

```
         }
    double *p = T+5; (0.75pt)
    for(int i=0; i<N-5; i++) (0.5pt)
    {cout << "p[" << i << "]=" << *p << endl;  (0.25pt)
    p++; } (0.25pt)
}
```

ou

```
main(){
    int N;   (0.25pt)
    cout << "N=" ;
    cin >> N; (0.25pt)
    double * T=new double[N]; (0.75pt)
    double *pt = T;
    for(int i=0; i<N; i++) (0.5pt)
        {
           cout << "T[" << i << "]=" ; cin >> *pt; (0.25pt)
           pt++; (0.25)
         }
    double *p = T+5; (0.75pt)
    cout << "Affichage du contenu du tableau à partir de la case 5 : " << endl;
    while(p<T+N){              (0.5pt)
      cout << *p<<" ";        (0.25pt)
      p++;                         (0.25pt)
     }
}
```

Exercice 3 : 2.25 pts

La suite du programme :

```
#include <iostream>
main(){
        char * jour[7] = {"lundi", "mardi", "mercredi", "jeudi", "vendredi", "samedi", "dimanche"};
        int i;
        cout << "Donnez un entier entre 1 et 7 :" ; cin >> i;
cout << "Le jour numéro " << i << "est : " << jour[i-1] << endl;
        char * c1 = jour[5]; (0.25pt)
        char * c2 = jour[6]; (0.25pt)
        for(int i=4; i>=0; i--)  jour[i+2] = jour[i];  (0.25*2pt)
jour[0]=c1; jour[2]=c2; (0.25*2pt)
       for(int i=0; i<7;i++) (0.5pt)
          {
            cout << "Le jour " << i << " = " << jour[i] << endl; (0.25pt)
          }
 }
```

Exercice 4 : 1.0 pt

(0.5+0.5pt)
A partir de l'arbre d'exécution, on peut conclure que *moris(1,0)* fait toujours appel à *moris(1,0)* d'une manière infinie ce qui montre que *moris(1,0)* ne se termine jamais.

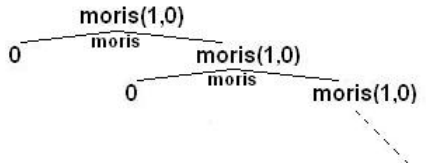

FIGURE 3 – L'arbre d'exécution de *moris(1,0)*