UNIVERSITE IBN TOFAIL ***Année: 2015/2016***
Faculté des sciences ***Filières : SMI***
Département d'Informatique ***Semestre : 3***
Kenitra

Algorithmique II
Examen final
Corrigé

Exercice 1 :(Sur 6 points)

1. <u>Déclarations</u>

```
Type Employe = Structure
        Matricule          : Caractere[1..10]
        Nom                : Caractere[1..15]
        Prénom             : Caractere[1..15]
        Sexe               : Caractere
        Date_Embauche      : Entier[1..3]
        Salaire            : Reel
Fin Structure
Var     E : Employe[1..50]
```

2. Algorithme Afficher(E : Employe[1..50])

```
    Var         i : Entier
                n : Entier      //pour sauvegarder le nombre d'employés qu'on cherche
    Début
        n←0
        Pour i←1 à 50 Faire
                Si E[i].Date_Embauche[3]=1980 Alors
                        Si (E[i].Salaire>=5000 et E[i].Salaire<=8000) Alors
                                n←n+1
                        Fin Si
                Fin Si
        Fin Pour
        Ecrire("\nLe nombre d'employés embauchés en 1980 et dont les salaires sont ")
        Ecrire("compris entre 5000 et 8000 est : ", n)
    Fin
```

Exercice 2 : (Sur 6 points)

1. <u>But de la fonction Chercher</u>

En analysant les instructions de la fonction Chercher on conclut qu'on est en train de chercher un couple (i,j), parmi tous les couples de {1,..,n}X{1,..,n}, tel que : A[i]+ B[j] =Som. Donc, la fonction Chercher retourne VRAI si ce couple existe ; sinon elle retourne FAUX.

2. <u>Complexités temporelles de la fonction Chercher</u>

- ***<u>Complexité dans les pires des cas</u> : Ce cas se présente lorsqu'on passe par tous les couples de {1,..,n}X{1,..,n}, sans trouver de couple (i,j) tel que A[i]+ B[j] =Som. Dans ce cas le nombre de passage dans la boucle "Tant que" est n2. D'où, si t1(n) la complexité temporelle dans les pires des cas de la fonction Chercher alors :***
  $t1(n)= 2taffect + n_{iter}(3tcomp+taddi+tcomp+2taffect+taddi)+3tcomp+taddi + tcomp+taddi+tretour.$
  Comme $n_{iter}=n^2$, alors
  $t1(n)=c0+c1*n^2=\theta(n^2)$ (où c0 et c1 sont des constantes)

- ***<u>Complexité dans les meilleurs des cas</u> : Ce cas se présente lorsqu'on trouve ce couple au premier passage dans la boucle "Tant que". D'où, si t2(n) est la complexité temporelle dans les meilleurs des cas de la fonction Chercher alors :***
  $t2(n)= 2taffect + n_{iter}(3tcomp+taddi+tcomp+2taffect+taddi)+3tcomp+taddi + tcomp+taddi+tretour$

Comme n_{iter}=1, alors t2(n)=C=θ(1) (0ù C est une constante)

Exercice 3

1. Fonction Somme_Rec

```
Fonction Somme_Rec(A : Entier[1.. n], k :Entier) : entier
Début
        Si (k <= 0) Alors      //Arrêt des appels quand le sous tableau est vide
                retourner(0)
        Sinon
                retourner(A[k]) +Somme_Rec(A, k− 1))
        Fin si
Fin
```

Au départ on appelle la fonction par Somme_Rec(A, n)

2.

a) Fonction Somme_Iter

```
Fonction Somme_Iter(A : Entier[1..n]) : Entier
Var     Somme : entier
Début
        Somme ← 0
        Pour i ← 1 à n Faire
                Somme ← Somme + A[i]
        Fin pour
        Retourner(Somme)
Fin
```

b) Preuve de validité de la fonction Somme_Iter

Pour prouver cet algorithme il faut démontrer deux choses : la terminaison et la validité de l'algorithme.

La terminaison de l'algorithme est triviale puisque la boucle est exécutée n fois et que le corps de la boucle n'est constitué que d'une somme et d'une affectation, opérations qui s'exécutent en un temps fini.

Pour démontrer la validité, nous allons considérer la propriété suivante : «À la fin de l'itération i, la variable Somme contient la somme des i premiers éléments du tableau A». Notons Sommei la valeur de la variable Somme à la fin de l'itération i et Somme0=0, sa valeur initiale. On effectue alors un raisonnement par récurrence sur i. La propriété est trivialement vraie pour i = 1 puisqu'à l'issue de la première itération Somme (initialisée à 0) contient la valeur A[1] :

Somme1 = Somme0 + A[1] = A[1]

Maintenant, si on suppose que la propriété est vraie pour une itération i alors :

$$\text{Somme} = \sum_{j=1}^{i} A[j]$$

À la fin de l'itération i + 1, Somme contiendra la valeur qu'elle contenait à la fin de l'itération i, donc la somme des i premiers éléments, plus A[i + 1] qui lui a été ajoutée à l'itération i + 1. Somme sera donc bien la somme des i + 1 premiers éléments de A :

$$\text{Somme}_{i+1} = \text{Somme}_{i} + A[i+1] = \sum_{j=1}^{i+1} A[j]$$

Vraie initialement, la propriété reste vraie à chaque nouvelle itération. Cette propriété est donc un invariant de l'algorithme. On parle d'invariant de boucle. L'algorithme se terminant, à la fin de l'itération n, il aura donc bien calculé la somme des n éléments du tableau A.