UNIVERSITE IBN TOFAIL **Année: 2017/2018**
Faculté des sciences **Filières : SMI**
Département d'Informatique **Semestre : 3**
Kenitra

Algorithmique II

Examen final

Durée : 1h 30mn

Exercice 1 : (Sur 7 points)

Un établissement universitaire organise les résultats finaux des étudiants d'une filière sous la forme d'un tableau *Filiere* composé de *N* éléments. Chaque élément du tableau est une structure contenant les champs suivants :

- ***Code Etudiant*** **qui est une chaine de 8 caractères**
- ***Nom*** **qui est une chaine de 15 caractères**
- ***Prénom*** **qui est une chaine de 15 caractères**
- ***Moyenne*** **qui est un nombre réel**

1. **Donner la déclaration du type *Etudiant* en tant que structure regroupant les champs ci-dessus et la déclaration de la variable *Filiere* en tant que tableau composé de N *Etudiant***
2. **Ecrire la procédure *Afficher Résultat*() qui permet d'afficher les résultats des étudiants de la filère comme suit :**

Numéro Code_Etudiant Nom Prénom Moyenne Mention

Où Numéro est le numéro d'ordre dans la liste, Code_Etudiant est le code de l'étudiant, Nom est le nom de l'étudiant, Prénom est le prénom de l'étudiant, Moyenne est la moyenne de l'étudiant et Mention est égale à :

"Ajourné" **si Moyenne < 10**
"Passable" **si 10 ≤ Moyenne < 12**
"A.Bien" **si 12 ≤ Moyenne < 14**
"Bien" **si 14 ≤ Moyenne < 16**
"T.Bien" **si 16 ≤ Moyenne**

Exercice 2 : (Sur 8 points)

On considère la fonction Calcul donnée par :

```
Fonction Calcul(A : Entier[1..n]) : Entier
Var     i, j : Entier
        Count, MaxCount :  Entier
Debut
        i ← 1
        j ← 1
        MaxCount ← 0
        Count ← 0
```

```
        Tant que  (i  <= n)  Faire
                Si (A[i] = A[j]) Alors
                        Count ← Count + 1
                Fin Si
                j ← j + 1
                Si  (j > n)  Alors
                        Si  (count > MaxCount)  Alors
                                MaxCount ← Count
                        Fin Si
                        Count ← 0
                        i ← i + 1
                        j ← i
                Fin Si
        Fin Tant Que
        Retourner  MaxCount
Fin     /*Fin de la fonction Calcul*/
```

1. **Expliquez brièvement ce que fait la fonction *Calcul***
2. **Déterminer la complexité temporelle dans le pire des cas de la fonction *Calcul*.**
3. **Ecrire une fonction *Meilleur()* faisant le même travail que la fonction *Calcul* et qui est de complexité temporelle, dans le pire des cas, strictement inférieure à celle de la fonction *Calcul* [indication : Vous pouvez exploiter un des algorithmes vus au cours, sans donner sa description].**

Exercice 3 : (Sur 5 points)

La fonction récursive d'Ackerman f est la fonction définie de INxIN dans IN par :

$$f(n,m) = \begin{cases} m+1 & Si\ \ n = 0 \\ f(n-1,1) & Si\ \ m = 0\ \ et\ \ n \geq 1 \\ f(n-1, f(n,m-1)) & Si\ \ n > 0\ \ et\ \ m > 0 \end{cases}$$

1. **Calculer f(1,0) et f(2,0)**
2. **Ecrire en pseudo_code la fonction récursive *Ackerman(n : Entier, m : Entier)* qui retourne la valeur de f(n,m)**