UNIVERSITE IBN TOFAIL
Faculté des sciences
Département d'Informatique
Kenitra

Année : 2021/2022
Filières : SMI
Semestre : 3

Algorithmique II

Examen final

Durée : 1h 30 mn

Exercice 1 : (Sur 7 points)

On rappelle qu'un nombre entier positif n>1 est dit <u>premier</u> si les seuls diviseurs de n sont 1 et n.
Un nombre entier positif n>1 est dit <u>semi-premier</u> si n est le produit de deux nombres premiers non nécessairement distincts.

1. **Ecrire une fonction *Premier(n : Entier)* qui retourne Vrai si n est premier, Faux sinon.**
2. **Ecrire une fonction *SemiPremier(n : Entier)* qui retourne Vrai si n est semi-premier, Faux sinon, ceci en utilisant la fonction *Premier*.**
3. **En utilisant la fonction *SemiPremier* définie ci-dessus, écrire l'algorithme *AfficheNombreSemiPremier,* qui détermine et affiche tous les nombres semi-premiers inférieurs à un entier N saisi par l'utilisateur.**

Exercice 2 : (Sur 6 points)

Étant donnés un texte *t* et un mot *m* sous forme de chaînes de caractères. La fonction suivante permet de déterminer le nombre de fois où le mot *m* apparaît dans le texte *t*.
Exemples :

- **le mot "elle" apparaît 2 fois dans le texte "quelle belle journee",**
- **le mot "aa" apparait 4 fois dans le texte "aaaaa".**

```
FONCTION chercherMot(m, t : chaine) : ENTIER
VAR     lt, lm, i, j, n, d : ENTIER
        Trouve : BOOLEEN
DEBUT
        lt ← longueur(t)        // longueur du texte
        lm ← longueur(m)        // longueur du mot
        n ← 0                   // nombre d'occurrences trouvées
        i ← 0                   // indice du caractère courant du texte
        d ← lt - lm+1           //différence plus 1 entre les longueurs lt et lm
        TANT QUE (i < d) FAIRE
                j ← 0   // indice du caractère courant du mot
                Trouve ← VRAI
                TANT QUE (j < lm ET Trouve) FAIRE
                        SI (t[i+j] <> m[j]) ALORS
                                Trouve ← FAUX
                        FIN SI
                        j ← j+1
                FIN TANT QUE
                SI Trouve ALORS
                        n ← n+1
                FIN SI
```

```
            i ← i+1
        FIN TANT QUE
        RETOURNER (n)
FIN
```

1. **Appliquer chercheMot(m, t) aux chaines t= "Quels bonbons !" et m="bon"**
2. **Déterminer la complexité *C(lm, lt)* en nombre de comparaisons de la fonction *chercherMot(m, t),* ceci en fonction de *lm* et *lt*, où *lm* et *lt* sont respectivement les longueurs des deux chaînes *m* et *t (on admet que les calculs de longueur(t) et longueur(m) ne contiennent pas d'opérations de comparaison) .***

Exercice 3 : (Sur 7 points)

On considère la fonction récursive *BinRecursive(n : Entier)*, donnée par :

```
Fonction BinRecursive(n : Entier) : chaine
Var     r : Entier
Début
        Si (n<0) Alors              //Si n<0 BinRecursive(n) retourne la chaine vide
                Retourner ""
        Fin Si
        Si (n<2) Alors
                Si (n=0) Alors
                        Retourner "0"
                Sinon
                        Retourner "1"
                Fin Si
        Sinon
                r ← n Mod 2
                Si (r=0) Alors
                        Retourner BinRecursive(n Div 2) + "0"
                Sinon
                        Retourner BinRecursive(n Div 2)+ "1"
                Fin Si
        Fin Si
Fin
```

1. **La récursivité de la fonction *BinRecursive(n)* est-elle terminale ou non terminale ? justifier votre réponse.**
2. **Donner les étapes d'exécution de *BinRecursive(13)*, ainsi que la chaine retournée par cet appel.**
3. **Sachant que la fonction *BinRecursive(n) retourne le codage binaire d'un entier positif,* écrire une fonction itérative *BinIterative(n)* équivalente à cette fonction.**