RÉPUBLIQUE TUNISIENNE MINISTÈRE DE L'ÉDUCATION ●●●●● **EXAMEN DU BACCALAURÉAT** SESSION **2018**	**Session principale**	
	Épreuve : **Algorithmique et Programmation**	Section : **Sciences de l'informatique**
	Durée : **3h**	Coefficient de l'épreuve : **2.25**

Section : .... N° d'inscription : .... Série : ....

Nom et prénom : ....

Date et lieu de naissance : ....

Signatures des surveillants

....

....

Le sujet comporte 5 pages numérotées de 1/5 à 5/5.
Cette feuille doit être remise à la fin de l'épreuve.

Exercice 1 : (3 points)

Dans un contexte informatique et pour chacune des propositions citées ci-dessous, mettre dans chaque case, la lettre **V** si la proposition est correcte ou la lettre **F** dans le cas contraire.

1. L'opération de décalage est utilisée dans :

- ☐ le tri rapide
- ☐ le tri insertion
- ☐ le tri Shell

2. Le tri insertion est un cas particulier :

- ☐ du tri sélection
- ☐ du tri à bulle
- ☐ du tri Shell

3. Le pas du tri Shell noté **P** est déterminé en utilisant la suite :

☐ $\begin{cases} P_0=0 \\ P_n=3+P_{n-1} \end{cases}$ ☐ $\begin{cases} P_0=1 \\ P_n=2*P_{n-1} \end{cases}$ ☐ $\begin{cases} P_0=1 \\ P_n=3*P_{n-1}+1 \end{cases}$

4. La fonction **Verif** permet de vérifier si les **N** entiers d'un tableau **T** sont triés en ordre croissant :

☐
```
0) Def FN Verif (T:Tab ;
   N:entier) : Booléen
1) Si (N=1) Alors Verif←Vrai
   Sinon
      Si (T[N]<T[N-1]) Alors
                Verif←Faux
      Sinon
        Verif←Fn Verif (T,N-1)
      Fin Si
2) Fin Verif
```

☐
```
0) Def FN Verif (T:Tab ;
   N:entier) : Booléen
1) Si (N=1) Alors
               Verif←Vrai
   Sinon
     Verif←(T[N]≥T[N-1])
        ET Fn Verif(T,N-1)
   Fin Si
2) Fin Verif
```

☐
```
0) Def FN Verif (T:Tab ;
   N:entier) : Booléen
1) Si (N=1) Alors Verif←Faux
   Sinon
     Si (T[N]<T[N-1]) Alors
          Verif←Vrai
     Sinon
       Verif←Fn Verif(T,N-1)
     Fin Si
2) Fin Verif
```

***Important* :**

Dans tout ce qui suit, chaque solution sous forme d'une analyse ou d'un algorithme doit être accompagnée d'un tableau de déclaration des objets ayant la forme suivante :

Objet	*Type/Nature*	*Rôle*

Exercice 2 : (3 points)

La suite de **Fibonacci** peut être définie comme suit :

$$\begin{cases} \mathbf{F_0 = 1} \\ \mathbf{F_1 = 1} \\ \text{Pour tout } \mathbf{n} \text{ pair, } \mathbf{F_n = (F_{p-1})^2 + (F_p)^2} & \text{avec } n = 2 * p \\ \text{Pour tout } \mathbf{n} \text{ impair, } \mathbf{F_n = (2*F_{p+1} - F_p)*F_p} & \text{avec } n = 2 * p + 1 \end{cases}$$

a) Ecrire un algorithme d'une fonction récursive nommée **Fibo** qui permet de calculer le terme $\mathbf{F_n}$ de la suite de **Fibonacci**, en utilisant la suite **F** décrite précédemment.

b) La formule $S = F_{n+2} - \mathbf{1}$ permet de calculer la somme S des **n+1** premiers termes de la suite de **Fibonacci** (de F_0 à F_n).
En utilisant cette formule et la fonction **Fibo**, écrire un algorithme d'une fonction nommée **Fibo_Som** qui permet de calculer la somme S.

Exercice 3 : (4 points)

Soit l'algorithme de la fonction **Inconnue** suivant :

```
0) DEF FN Inconnue (E, k : entier) : booléen
1) SI (E < 2) OU (E mod k = 0) Alors Inconnue ← Faux
      Sinon Si  k > Racine_carrée(E) Alors  Inconnue ← Vrai
      Sinon  Inconnue ← FN Inconnue (E, k+1)
      FinSi
2) Fin Inconnue
```

1. Déterminer la valeur retournée par la fonction **Inconnue** pour chacun des quatre appels suivants :
   - Inconnue (5,2)
   - Inconnue (6,2)
   - Inconnue (7,2)
   - Inconnue (9,2)
2. Déduire le rôle de cette fonction.
3. Ecrire un algorithme d'une fonction **Calcul (epsilon)** permettant de retourner une valeur approchée de $\boldsymbol{\pi}$ à epsilon près, en utilisant la formule de **Zêta Riemann** suivante :

$$\frac{\pi^2}{6} = \frac{2^2}{(2^2-1)} * \frac{3^2}{(3^2-1)} * \frac{5^2}{(5^2-1)} * \frac{7^2}{(7^2-1)} * \frac{11^2}{(11^2-1)} * \frac{13^2}{(13^2-1)} * \dots * \frac{P^2}{(P^2-1)}$$

Avec **P** un entier tel que **Inconnue (P, 2) = Vrai**

Problème : (10 points)

On se propose de simuler la multiplication de deux entiers naturels **A** et **B** (**avec A et B dans [10, 10000]**), en utilisant la méthode du mathématicien **Ibn Al Banna** comme suit :

a) Former deux chaînes **CA** et **CB** contenant respectivement, les chiffres de l'entier **A** et les chiffres de l'entier **B**.

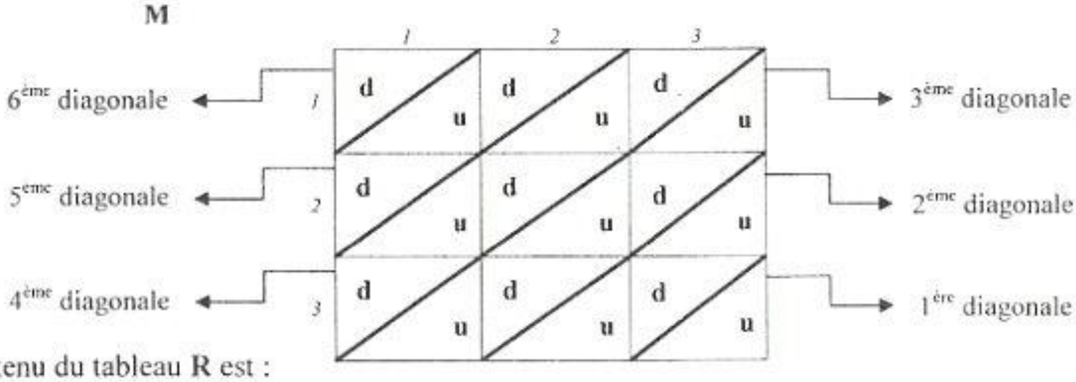
b) Ajuster la longueur des deux chaînes **CA** et **CB** en complétant par des zéros à gauche, si nécessaire, l'une des deux chaînes pour qu'elles soient de même longueur. **CA** et **CB** seront formées chacune de **n** chiffres et auront la forme suivante :

CA = "A_n … $A_4A_3A_2A_1$"
CB = "B_n … $B_4B_3B_2B_1$"

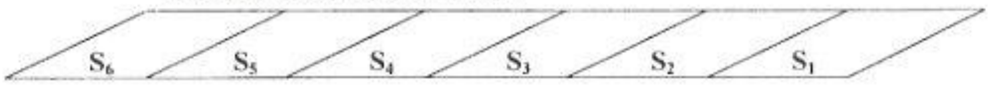
c) Générer une matrice carrée **M** de taille **nxn**, tel que :

M[i,j] = le produit du i^{ème} chiffre de la chaîne CA par le j^{ème} chiffre de la chaîne CB

d) La matrice **M** est supposée formée par **2xn** diagonales en plaçant le chiffre des dizaines dans la moitié supérieure de chaque case et le chiffre des unités dans la moitié inférieure de la même case, comme le montre la figure ci-après pour le cas où **n**=3.
Remplir chacune des cases d'un tableau **R** de taille **2xn** par la somme des chiffres d'une diagonale de la matrice **M** en commençant par la diagonale du coin inférieur droit jusqu'à celle du coin supérieur gauche.



Le contenu du tableau **R** est :



Avec :

- **d** et **u** sont respectivement, le chiffre des dizaines et le chiffre des unités de chaque élément de la matrice.
- S_i est la somme des chiffres de la diagonale n° **i** .

e) En commençant par la dernière case du tableau **R** et pour chaque élément **R[i]** supérieur ou égal à **10**, mettre à jour son contenu comme suit :

$$\begin{cases} R[i-1] = R[i-1] + R[i] \text{ Div } 10 \\ R[i] = R[i] \text{ Mod } 10 \end{cases}$$

f) Le résultat du produit des deux entiers **A** et **B** est obtenu en concaténant de gauche à droite les éléments du tableau **R** obtenus dans l'étape précédente.

Exemple : Pour **A = 7842** et **B = 35**.

a) Les deux chaines **CA** et **CB** seront respectivement **"7842"** et **"35"**.

b) Après ajustement des longueurs des deux chaînes, on aura : CA="7842", CB="0035" et n = 4.

c) La matrice carrée **M** sera :

Le produit du 1[er] chiffre de **CA** (7) par le 1[er] chiffre de **CB** (0) est 0

Le produit du 2[ème] chiffre de **CA** (8) par le 3[ème] chiffre de **CB** (3) est 24

	1	2	3	4
1	**0**	**0**	**21**	**35**
2	**0**	**0**	**24**	**40**
3	**0**	**0**	**12**	**20**
4	**0**	**0**	**6**	**10**

Le produit du 3[ème] chiffre de **CA** (4) par le 4[ème] chiffre de **CB** (5) est 20

d) Génération du tableau **R** :

En supposant que la matrice **M** est formée par **2xn** diagonales, son contenu sera représenté comme suit :

	1	2	3	4
1	**0** / **0**	**0** / **0**	**2** / **1**	**3** / **5**
2	**0** / **0**	**0** / **0**	**2** / **4**	**4** / **0**
3	**0** / **0**	**0** / **0**	**1** / **2**	**2** / **0**
4	**0** / **0**	**0** / **0**	**0** / **6**	**1** / **0**

Le contenu du tableau **R** est :

0	0	2	6	14	4	7	0

En effet, les sommes sont :

- ✓ S1 = 0
- ✓ S2 = 0+1+6 = 7
- ✓ S3 = 0+2+2+0+0 = 4
- ✓ S4 = 5+4+4+1+0+0+0 = 14
- ✓ S5 = 3+1+2+0+0+0+0 = 6
- ✓ S6 = 2+0+0+0+0 = 2
- ✓ S7 = 0+0+0 = 0
- ✓ S8 = 0

e) La mise à jour du tableau **R** donne :

0	0	2	7	4	4	7	0

f) La concaténation des éléments du tableau **R** donne le résultat : **00274470**

Ci après, on propose l'algorithme **Diag** d'un module permettant de remplir le tableau **R** par les sommes des diagonales de la matrice **M** et de le mettre à jour comme décrit précédemment dans les deux étapes **d)** et **e)** :

```
0) Def Proc Diag (M: Matrice; N: Octet; Var R: Tab)
1) Pour k de 2*N à N (Pas = -1) Faire
        i←N, j←k-i, R[k]←M[i,j] mod 10
        Tantque (j < N) Faire
                j←j+1
                R[k] ← R[k]+M[i,j] div 10 + M[i-1,j] mod 10
                i←i-1
        Fin Tantque
   Fin Pour
   Pour k de N à 1 (Pas = -1) Faire
        R[k]←M[k,1] div 10, i←k, j←1
        Tantque (i > 1) Faire
                i←i-1
                R[k]← R[k]+M[i,j] mod 10 + M[i,j+1] div 10
                j←j+1
        Fin Tantque
   Fin Pour
2) Proc MiseAjour (R, N)
3) Fin Diag
```

Travail demandé :

1. Proposer les déclarations des nouveaux types **Matrice** et **Tab** utilisés dans l'algorithme du module **Diag**.
2. Développer un algorithme pour la procédure **MiseAjour (R,N)** qui permet de mettre à jour le tableau R comme expliqué précédemment.
3. Analyser le problème en le décomposant en modules et en utilisant le module **Diag**.

   NB : Prévoir la saisie des entiers **A** et **B** (**avec A et B dans [10, 10000]**) et l'affichage du résultat.
4. Ecrire les algorithmes des modules envisagés.