Bac informatique

Page No.

RÉPUBLIQUE TUNISIENNE MINISTÈRE DE L'ÉDUCATION EXAMEN DU BACCALAURÉAT SESSION 2020	**Session de contrôle**	
	Épreuve : **Algorithmique et Programmation**	Section : **Sciences de l'informatique**
	Durée : **3h**	Coefficient de l'épreuve : **2.25**

ꕥ ꕥ ꕥ ꕥ ꕥ ꕥ

Le sujet comporte 4 pages numérotées de 1/4 à 4/4.

Important :

Chaque solution développée par le candidat sous forme d'un algorithme doit être accompagnée d'un tableau de déclaration des objets ayant la forme suivante :

Objet	*Type / Nature*	*Rôle*

Exercice 1 : (3 points)

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

```
0) Def Fn Inconnue (Var F : Fiche) : Entier
1) Ouvrir (F)
   i ← 0
   j ← Taille_fichier (F) – 1
   Lire (F, e1)
   Pointer (F, j)
   Lire (F, e2)
   Tant que (i < j) faire
        Si e1 > e2 Alors
                j ← j – 1
                Pointer (F, j)
                Lire (F, e2)
        Sinon
              i ← i+1
              Pointer (F, i)
              Lire (F, e1)
        FinSi
   Fin Tant que
2) Pointer (F, i)
   Lire (F, e1)
3) Inconnue ← e1
4) Fin Inconnue
```

Travail demandé :

1) Dresser le tableau de déclaration des objets locaux de la fonction **Inconnue**.
2) Dresser le tableau de déclaration du type **Fiche**.
3) Donner la valeur retournée par la fonction **Inconnue** pour le fichier **F** contenant les valeurs suivantes :

F = 5 11 3 7 18 13 8

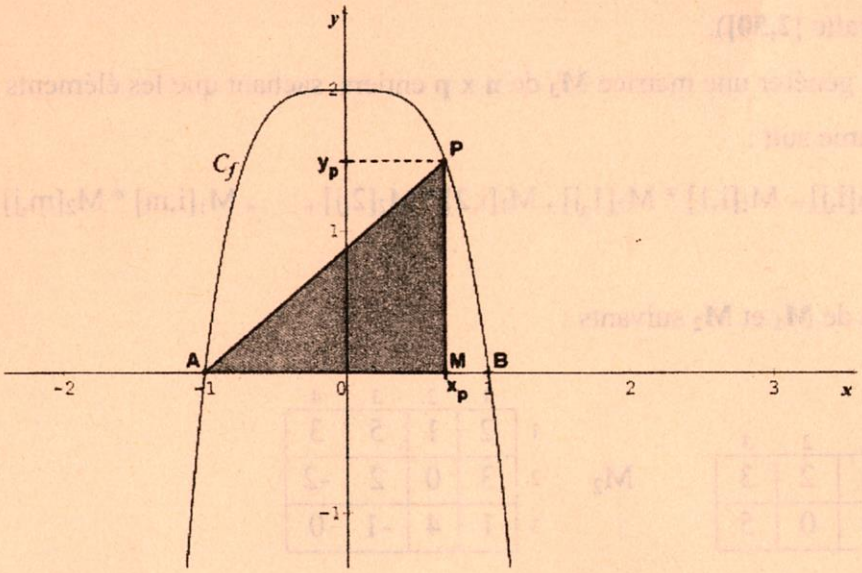
4) Parmi les quatre rôles ci-dessous, réécrire sur votre feuille de copie celui qui correspond au rôle de la fonction **Inconnue**.

- La fonction **Inconnue** détermine le maximum parmi les éléments d'un tableau.
- La fonction **Inconnue** détermine le minimum parmi les éléments d'un tableau.
- La fonction **Inconnue** détermine le maximum parmi les éléments d'un fichier.
- La fonction **Inconnue** détermine le minimum parmi les éléments d'un fichier.

5) Modifier la séquence d'instructions **2** pour que la fonction **Inconnue** retourne la position de **e1** dans **F**.

Exercice 2 : (3 points)

Soit la fonction $f(x) = -2 * x^4 + 2$. La figure ci-après représente sa courbe ***Cf*** :



La courbe ***Cf*** coupe l'axe des abscisses en deux points **A** et **B** de coordonnées respectivement **(-1, 0)** et **(1, 0)**.

Soient **P** un point de la courbe ***Cf*** situé entre **A** et **B** de coordonnées (x_p, y_p) et **M** le point de coordonnées $(x_p,0)$. Le triangle **AMP** est un triangle rectangle en **M** (triangle grisé dans la figure).

Travail demandé :

Ecrire un module nommé **Aire_triang** qui permet de déterminer une valeur approchée de l'abscisse x_p du point **P** à 10^{-5} près pour que l'aire du triangle rectangle **AMP** soit maximale.

N.B. : On rappelle que l'aire du triangle **AMP** est égale à $(x_p - x_A) * f(x_p)/2 = (x_p + 1) * f(x_p)/2$

Avec x_p l'abscisse du point **P** et x_A l'abscisse du point **A**.

Voir suite au verso ☞

Exercice 3 : (4 points)

Soient **n** un entier naturel et **U** une suite arithmétique définie par :

$$\begin{cases} U_0 = 1 \\ U_{n+1} = \dfrac{(-2)^{n+1}}{2 * U_n} \end{cases}$$

1) Quel est l'ordre de récurrence de la suite **U** ? Justifiez votre réponse.
2) Ecrire un algorithme d'un module nommé **Suite** qui permet de calculer le terme U_n pour tout entier naturel **n**.

   N.B. : L'entier **n** est saisi dans le programme appelant.

Exercice N°4 : (3,5 points)

Soient M_1 une matrice de **n** x **m** entiers et M_2 une matrice de **m** x **p** entiers (avec **n**, **m** et **p** trois entiers de l'intervalle **[2,50]**).

On se propose de générer une matrice M_3 de **n** x **p** entiers, sachant que les éléments de cette matrice sont calculés comme suit :

$$M_3[i,j] = M_1[i,1] * M_2[1,j] + M_1[i,2] * M_2[2,j] + \ldots\ldots + M_1[i,m] * M_2[m,j]$$

Exemple :

Pour les éléments de M_1 et M_2 suivants :

M_1

	1	2	3
1	1	2	3
2	4	0	5

M_2

	1	2	3	4
1	2	1	5	3
2	3	0	2	-2
3	1	4	-1	0

La matrice M_3 sera :

M_3

	1	2	3	4
1	11	13	6	-1
2	13	24	15	12

En effet,

- $M_3[1,1] = M_1[1,1]*M_2[1,1] + M_1[1,2]*M_2[2,1] + M_1[1,3]*M_2[3,1] = 1*2+2*3+3*1 = 11$
- $M_3[2,1] = M_1[2,1]*M_2[1,1] + M_1[2,2]*M_2[2,1] + M_1[2,3]*M_2[3,1] = 4*2+0*3+5*1 = 13$
- ...
- $M_3[2,4] = M_1[2,1]*M_2[1,4] + M_1[2,2]*M_2[2,4] + M_1[2,3]*M_2[3,4] = 4*3+0*(-2)+5*0 = 12$

Travail demandé :

1) Dresser un tableau de déclaration d'un type pour les matrices M_1, M_2 et M_3.
2) Ecrire un algorithme d'un module **Prod_Mat (M_1, M_2, M_3, n, m, p)** qui permet de générer une matrice M_3 de **n** x **p** entiers à partir des deux matrices M_1 et M_2 respectivement de **n** x **m** et de **m** x **p** entiers, en appliquant le procédé décrit précédemment.
   N.B. : M_1, M_2, **n**, **m** et **p** sont saisis dans le programme appelant.

Exercice 5 : (6,5 points)

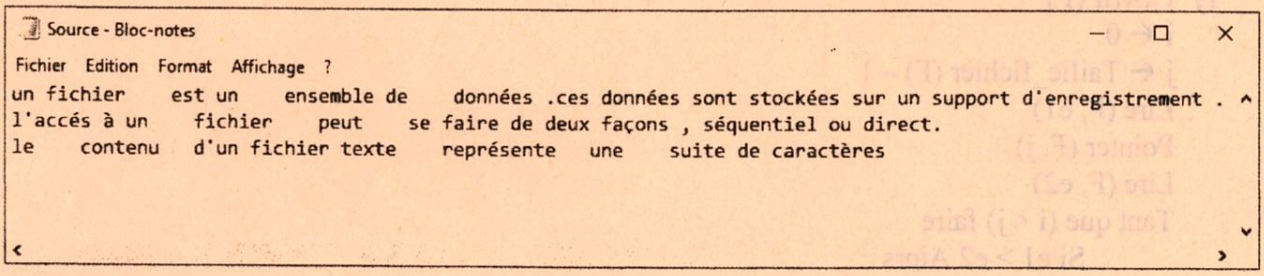
On se propose de nettoyer un fichier texte "**Source.txt**" pour générer un fichier "**Resultat.txt**", en respectant les règles suivantes :

- Le texte ne doit pas comporter des espaces successifs ;
- Si une ligne du texte commence par une lettre, cette dernière doit être en majuscule ;
- Avant un point ou une virgule il n'y a pas d'espace ;
- Après une virgule, il doit y avoir un espace ;
- Après un point, il doit y avoir un espace et s'il est suivi d'une lettre elle doit être en majuscule, à l'exception du point qui peut se trouver à la fin d'une ligne.

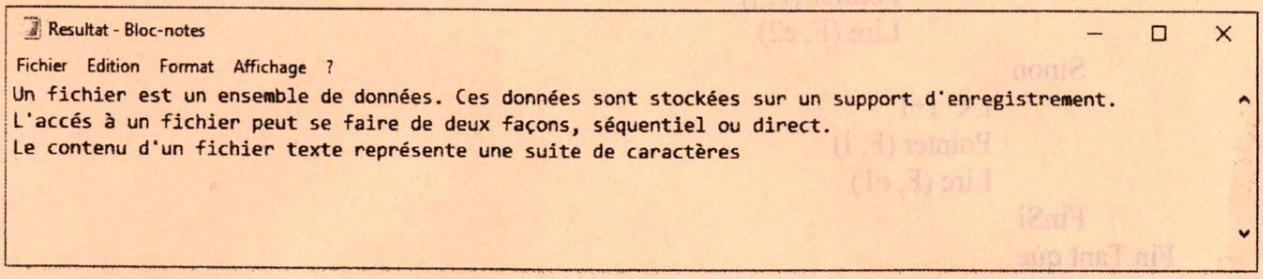
N.B. : Chaque ligne du fichier est composée d'au maximum 255 caractères.

Exemple :

Pour le fichier "**Source.txt**" suivant :



Après nettoyage des lignes du fichier "**Source.txt**", le fichier "**Resultat.txt**" sera :



Travail demandé :

1) Donner une instruction d'association pour chacun des deux fichiers "**Source.txt**" et "**Resultat.txt**" respectivement aux variables logiques **S** et **R**, sachant que les deux fichiers sont enregistrés sur la racine du disque **D**.
2) Ecrire un module nommé **Nettoi_F** qui permet à partir d'un fichier "**Source.txt**" déjà saisi dans le programme appelant, de créer et de générer un deuxième fichier "**Resultat.txt**" en respectant les règles décrites précédemment.

RÉPUBLIQUE TUNISIENNE **MINISTÈRE DE L'ÉDUCATION** **EXAMEN DU BACCALAURÉAT** SESSION **2019**	**Session principale**	
	Épreuve : **Algorithmique et Programmation**	Section : **Sciences de l'informatique**
	Durée : **3h**	Coefficient de l'épreuve : **2.25**

Le sujet comporte 4 pages numérotées de 1/4 à 4/4.

Important :

Chaque solution développée par le candidat, sous forme d'une analyse ou d'un algorithme doit être accompagnée d'un tableau de déclarations des objets ayant la forme suivante :

Objet	*Type / Nature*	*Rôle*

Exercice 1 (2 points)

Soit l'algorithme suivant de la fonction récursive intitulée **Quoi** :

```
0) DEF FN Quoi (a, b : Réel) : ……
1) Si (a – b) ≥ 0 Alors
       Quoi ← a
   Sinon Quoi ← FN Quoi (b, a)
   Fin Si
2) Fin Quoi
```

Travail demandé :

Reproduire le tableau suivant, puis en se référant à l'algorithme de la fonction **Quoi** et pour chacune des propositions ci-après, remplir la case correspondante par la lettre de la réponse correcte.

Proposition	**1**	**2**	**3**	**4**
Réponse				

1. Le type de la fonction **Quoi** peut être :
   - **a)** Octet
   - **b)** Réel
   - **c)** Entier long
2. La condition d'arrêt du traitement récursif est :
   - **a)** (a – b) ≥ 0
   - **b)** Quoi ← a
   - **c)** Quoi ← FN Quoi (b, a)
3. Pour a = 9 et b = 12, le résultat retourné par la fonction **Quoi** est égal à :
   - **a)** 9
   - **b)** 12
   - **c)** 3
4. Le rôle de la fonction **Quoi** est de :
   - **a)** calculer le PPCM de a et b
   - **b)** calculer le PGCD de a et b
   - **c)** rechercher le maximum de a et b

Exercice 2 (2,75 points)

La figure ci-dessous représente la courbe de la fonction **f** définie par $f(x) = \frac{1}{x}$ sur l'intervalle $]0, +\infty[$.

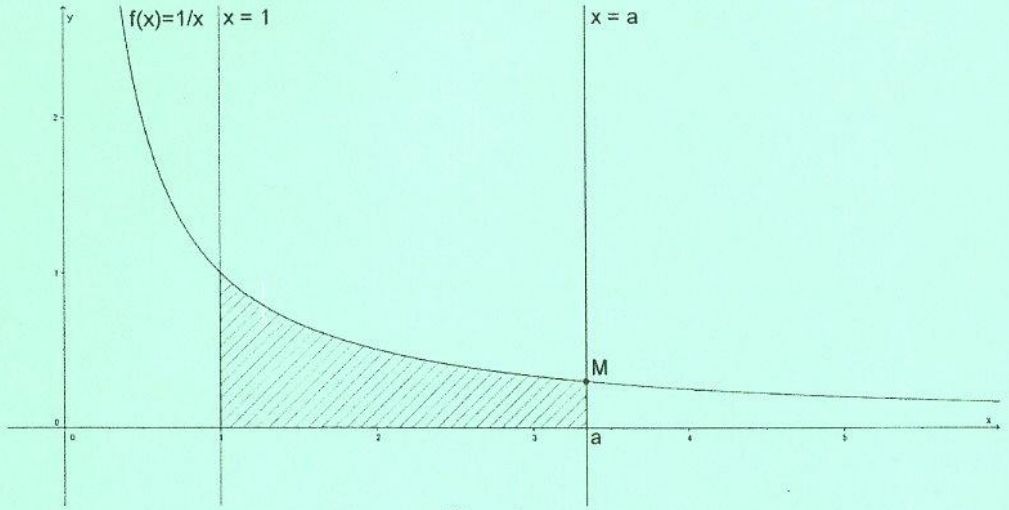


Figure 1

Etant donné que $\int_1^a \frac{1}{x} dx \begin{cases} < 1 \ si \ \ a < e \\ = 1 \ si \ \ a = e \\ > 1 \ si \ \ a > e \end{cases}$, on remarque que la surface délimitée par les deux droites d'équations x = 1 et x = a, l'axe des abscisses et la courbe f(x), varie selon la valeur de l'abscisse **a** du point M (la surface hachurée dans la **Figure1**). Cette surface sera égale à **1** lorsque la valeur de **a** est égale au nombre d'Euler **e**.

Travail demandé :

1. Ci-dessous une partie d'un algorithme de la fonction **Surface**, qui permet de calculer la surface hachurée en fonction de l'abscisse **a** du point M et en utilisant la méthode des rectangles à gauche.

0) DEF FN Surface (a : réel ; n : entier) : réel
1) S ← 0
……
h ← (a-1) / n
Pour i de 1 à n Faire
……
x ← x + h
Fin Pour
2) ……
3) Fin Surface

T.D.O		
Objet	**Type**	**Rôle**
S	Réel	Calculer la somme des f(x)
x	Réel	Contenir les valeurs des abscisses
h	Réel	Contenir la largeur des rectangles
i	Entier	Compteur

NB : n représente le nombre de rectangles.

Réécrire l'algorithme de la fonction **Surface** en complétant les vides par les trois instructions convenables à partir de la liste d'instructions suivante :

x ← 1	S ← S + 1/x	Surface ← S*h
x ← 0	S ← S + ½(1/x+1/(x+h))	Surface ← S*n*h

2. En faisant appel à la fonction **Surface**, écrire un algorithme d'une fonction **Calcul (a, n)** permettant de déterminer une valeur approchée du nombre d'Euler **e**, qui correspond à une valeur de la surface proche de **1** avec une précision de 10^{-4}.

NB : On pourra calculer le nombre d'Euler **e** en variant l'abscisse **a** par pas de 10^{-4}.

Exercice 3 (5,25 points)

Une **fraction** de la forme $\frac{a}{b}$ est dite **irréductible** lorsqu'on ne peut plus la simplifier.

Mathématiquement, une fraction $\frac{a}{b}$ est irréductible si le **PGCD (a, b) = 1**.

Ainsi, pour rendre une fraction $\frac{a}{b}$ irréductible, on divise le numérateur et le dénominateur de cette fraction par le **PGCD (a, b)**.

Soit "**Fraction.dat**" un fichier d'enregistrements contenant des fractions représentée chacune par les deux champs suivants :

- **Num** : un entier représentant le numérateur de la fraction.
- **Denom** : un entier représentant le dénominateur de la fraction.

On se propose de créer puis d'afficher, un fichier d'enregistrements intitulé "**Irreduct.dat**" qui devra contenir pour chaque fraction du fichier "**Fraction.dat**" la fraction irréductible correspondante.

Travail demandé :

1. Ecrire un algorithme d'un module permettant de remplir puis d'afficher le fichier "**Irreduct.dat**" comme expliqué ci-dessus.

   NB :

   - Le candidat n'est pas appelé à remplir le fichier "**Fraction.dat**".
   - Les fichiers "**Fraction.dat**" et "**Irreduct.dat**" ont la même structure.
   - Il est possible d'utiliser la méthode de la **division euclidienne** pour calculer le **PGCD** de deux entiers **a** et **b** dont le principe se présente comme suit :
     - diviser **a** par **b** pour obtenir un reste **r**,
     - si **r = 0**, le **PGCD** est égal à **b**,
     - si **r ≠ 0**, refaire la division en remplaçant **a** par **b** et **b** par **r** jusqu'à obtenir **r = 0**. Dans ce cas le **PGCD** est égal au dernier reste non nul.

2. Donner une déclaration pour chaque nouveau type utilisé dans la réponse à la question **1**.

Problème (10 points)

Parmi les méthodes de chiffrement utilisant un mot-clé, on cite celle décrite ci-après qui permet de crypter un message **msg** ne dépassant pas **18** caractères et formé uniquement de lettres minuscules, de chiffres et d'espaces :

Etape1 : Remplir aléatoirement une matrice carrée **M1** de dimension **6x6** par toutes les lettres alphabétiques minuscules ainsi que tous les chiffres.

NB : Les indices des lignes et des colonnes de la matrice **M1** sont les lettres **A, B, C, D, E** et **F**.

Etape2 : Générer un message intermédiaire **msgi**, en concaténant les résultats du chiffrement de chaque caractère du message **msg**. Le résultat du chiffrement d'un caractère est la concaténation de l'indice de la ligne avec l'indice de la colonne de la case contenant le caractère à chiffrer.
Le caractère espace ne sera pas chiffré.

Etape3 : Remplir une deuxième matrice **M2** de taille **7x6** caractères en mettant dans :

- la première ligne, les lettres d'un mot-clé formé de **6** lettres majuscules,
- le reste des lignes, le message **msgi** caractère par caractère en commençant par la première case de la deuxième ligne.

NB : Chaque case vide de la matrice **M2** sera remplie par le caractère espace.

Etape4 : Trier les éléments de la 1ère ligne de **M2** selon un ordre alphabétique croissant, sachant que tout déplacement d'un élément entraîne le déplacement de tous les éléments de la colonne correspondante.

Etape5 : Concaténer les lettres de la matrice **M2**, colonne par colonne en commençant par la 1ère colonne et sans considérer les éléments de la 1ère ligne, pour obtenir le message chiffré final.

Exemple :

Pour **msg** = "**promotion bac 2019**" et le mot-clé "**CHAISE**"

Etape1 : La matrice **M1** est remplie aléatoirement comme suit :

	A	B	C	D	E	F
A	c	1	o	f	w	j
B	y	m	t	5	b	4
C	i	7	a	2	8	s
D	p	3	0	q	h	x
E	k	e	u	l	6	d
F	v	r	g	z	n	9

Etape2 : Le résultat du chiffrement du message **msg**, caractère par caractère donne :

Caractère à chiffrer	p	r	o	m	o	t	i	o	n		b	a	c		2	0	1	9
Résultat du chiffrement	DA	FB	AC	BB	AC	BC	CA	AC	FE		BE	CC	AA		CD	DC	AB	FF

D'où le message **msgi** est le suivant : "DAFBACBBACBCCAACFE BECCAA CDDCABFF"

Etape3 : Remplissage de la matrice **M2**

C	H	A	I	S	E
D	A	F	B	A	C
B	B	A	C	B	C
C	A	A	C	F	E
	B	E	C	C	A
A		C	D	D	C
A	B	F	F		

Etape4 : Tri de la matrice **M2**

A	C	E	H	I	S
F	D	C	A	B	A
A	B	C	B	C	B
A	C	E	A	C	F
E		A	B	C	C
C	A	C		D	D
F	A		B	F	

Etape5 :

Le message chiffré final est "**FAAECFDBC AACCEAC ABAB BBCCCDFABFCD** "

On se propose d'écrire un programme permettant :

- de saisir un message **msg** ne dépassant pas **18** caractères et formé uniquement de lettres minuscules, de chiffres et d'espaces.
- de saisir un mot-clé formé de **6** lettres majuscules.
- de crypter le message **msg** selon la méthode de chiffrement décrite précédemment.
  NB : On dispose d'un module **Initialisation(M1)** qui permet de remplir aléatoirement la matrice carrée **M1** comme décrit dans l'étape 1 et que le candidat peut appeler dans sa solution sans le développer.
- d'afficher le message chiffré final.

Travail demandé :

1. Analyser le problème en le décomposant en modules.
2. Ecrire les algorithmes des modules envisagés.

RÉPUBLIQUE TUNISIENNE **MINISTÈRE DE L'ÉDUCATION** **EXAMEN DU BACCALAURÉAT** SESSION **2019**	**Session de contrôle**	
	Épreuve : **Algorithmique et Programmation**	Section : **Sciences de l'informatique**
	Durée : **3h**	Coefficient de l'épreuve : **2.25**

Le sujet comporte 4 pages numérotées de 1/4 à 4/4.

Important :

Chaque solution développée par le candidat, sous forme d'une analyse ou d'un algorithme doit être accompagnée d'un tableau de déclarations des objets ayant la forme suivante :

Objet	*Type / Nature*	*Rôle*

Exercice 1 (2 points)

Reproduire le tableau suivant, puis pour chacune des propositions ci-après, remplir la case correspondante par la lettre de la réponse correcte.

Proposition	**1**	**2**	**3**	**4**
Réponse				

1. Dans un algorithme récursif, la condition d'arrêt permet :
   - **a)** de changer la valeur d'un paramètre.
   - **b)** de stopper les appels récursifs.
   - **c)** d'appeler le module autant de fois, dans son corps.

2. Le processus de partitionnement dans la méthode du tri rapide permet de :
   - **a)** diviser un tableau en trois parties égales.
   - **b)** fusionner deux tableaux triés.
   - **c)** mettre tous les éléments inférieurs au pivot à sa gauche et les autres à sa droite.

3. L'algorithme qui permet de calculer la factorielle d'un nombre est :

a)

```
0) DEF FN Fact (n:octet) : entier
1) Si n = 0 Alors Fact ← n
   Sinon Fact ← n * FN Fact (n-1)
   Fin Si
2) Fin Fact
```

b)

```
0) DEF FN Fact (n:octet) : entier
1) Si n > 1
   Alors Fact ← n * FN Fact (n-1)
   Fin Si
2) Fin Fact
```

c)

```
0) DEF FN Fact (n:octet) : entier
1) F ← 1
   Pour i de 2 à n faire
      F ← F * i
   Fin Pour
2) Fact ← F
3) Fin Fact
```

4. Dans le problème des tours de Hanoï, le nombre minimal de déplacements de n disques nécessaire pour arriver à l'état final est :
   - **a)** $2^n - 1$
   - **b)** 2^{n-1}
   - **c)** $2^n + 1$

Exercice 2 (3 points)

Soit l'algorithme suivant de la fonction intitulée **Quoi** :

```
0) DEF FN Quoi (T : Tab ; a, deb, fin : entier) : ……
1) m ← (deb + fin) Div 2
2) Si (deb > fin) Alors Quoi ← faux
     Sinon Si (T[m] = a) Alors Quoi ← vrai
       Sinon Si (T[m] > a) Alors Quoï ← FN Quoi (T, a, deb, m-1)
         Sinon Quoi ← FN Quoi (T, a, m+1, fin)
   Fin Si
3) Fin Quoi
```

NB : T est un tableau d'entiers trié dans l'ordre croissant.

Travail demandé :

1- Donner le type de retour de la fonction **Quoi**.

2- Donner la valeur retournée par la fonction **Quoi** (T, 6, 1, 5) pour les deux cas suivants :

Cas 1 :

T	-2	3	6	10	12

Cas 2 :

T	-2	3	4	10	12

3- Déduire le rôle de la fonction **Quoi**.

Exercice 3 (4 points)

Le jeu **des tours de Hanoï** est un jeu de réflexion qui consiste à déplacer **N** disques de diamètres différents d'une tour de départ (**D**) à une tour d'arrivée (**A**) en passant par une tour intermédiaire (**Inter**), et ceci en un minimum de coups, tout en respectant les règles suivantes :

- On ne peut pas déplacer plus d'un disque à la fois.
- On ne peut placer un disque que sur un autre disque plus grand ou sur un emplacement vide.

Soit l'algorithme récursif suivant de la procédure qui simule le jeu des tours de Hanoï :

```
0) DEF PROC Hanoi (N : entier ; D, A, Inter : caractère)
1) Si (N > 0) Alors
   PROC Hanoi (N-1, D, Inter, A)
   Ecrire ("Déplacer un disque de ", D, " à ", A)
   PROC Hanoi (N-1, Inter, A, D)
   Fin Si
2) Fin Hanoi
```

Travail demandé :

1) Pour sauvegarder les différents déplacements des **N** disques, on se propose d'utiliser un fichier d'enregistrements intitulé "**Deplace.dat**".

   a) Proposer une déclaration d'un type enregistrement permettant de représenter un déplacement , sachant que chaque déplacement est caractérisé par une tour de départ et une tour d'arrivée.

   b) Proposer une déclaration d'un type pour le fichier "**Deplace.dat**".

2) Réécrire l'algorithme de la procédure **Hanoi** en apportant les modifications nécessaires pour qu'il procède à la sauvegarde des différents déplacements des disques dans le fichier d'enregistrements "**Deplace.dat**".
   NB : Le fichier résultat "**Deplace.dat**" est créé et ouvert dans le programme appelant.

3) Ecrire un algorithme d'un module permettant d'afficher le contenu du fichier d'enregistrements "**Deplace.dat**".

Problème (11 points)

Parmi les méthodes de cryptage utilisées pour crypter un texte **msg** formé de caractères alphabétiques, on cite la méthode **OU exclusif simple** utilisant une **clef** de cryptage, qui est une chaîne binaire représentée sur **N** bits avec **N** est un multiple de 7.

Cette méthode consiste à réaliser les étapes suivantes :

Etape1 :

Sachant que chaque caractère de **msg** est représenté sur 7 bits, on procède à un ajout du caractère antislash "\" à la fin de **msg**, autant de fois que nécessaire, jusqu'à ce que la longueur de sa représentation binaire devienne un multiple de **N**.

Etape2 :

On découpe **msg** en blocs de caractères pour que chacun soit représenté sur un nombre de bits égal au nombre de bits de la clef, c'est-à-dire que chaque bloc contient **N Div 7** caractères.

Etape3 :

On détermine la représentation binaire de chaque bloc par la concaténation de l'équivalent binaire, sur 7 bits, du code Ascii de chaque caractère du bloc.

Etape4 :

A chaque bloc, on applique un **Ou exclusif (OUex)** avec la clef, bit par bit.
On rappelle que **0 OUex 0 = 0, 1 OUex 0 = 1** et **1 OUex 1 = 0**

Etape5 :

Le texte crypté est obtenu en concaténant respectivement les résultats trouvés dans l'étape précédente.

Exemple **:**
Pour **clef** = "10101100101101" et **msg** = "Savon"

Etape1 :

Puisque le texte à crypter est formé de 5 caractères et chaque caractère est représenté sur 7 bits, la longueur de la représentation binaire du texte est 35 (5 caractères * 7 bits).

Comme la longueur de la clef est égale à 14 et comme 35 n'est pas un multiple de 14, il est nécessaire d'ajouter à droite du texte le caractère antislash "\" une seule fois afin que la nouvelle longueur (6*7=42) devienne un multiple de 14.

Etape2 :

Etant donné le texte modifié (**msg** = "Savon\"), on procède à un découpage en blocs de 14 bits c'est à dire en blocs de deux caractères (14 Div 7).

D'où, Bloc1 = "Sa", Bloc2 = "vo" et Bloc3 = "n\"

Etape3 :

La représentation binaire de chaque bloc est obtenue par la concaténation de l'équivalent binaire, sur 7 bits, du code Ascii de chaque caractère.

La représentation binaire du Bloc1 est "1010011 1100001"
S a

La représentation binaire du Bloc2 est "1110110 1101111"
v o

La représentation binaire du Bloc3 est "1101110 1011100"
n \

Etape4 :

En appliquant à chaque bloc la fonction **Ou exclusif** avec la clef, on obtient les résultats suivants :

Bloc1 **OUex** clef = 10100111100001 **OUex** 10101100101101 = 00001011001100
Bloc2 **OUex** clef = 11101101101111 **OUex** 10101100101101 = 01000001000010
Bloc3 **OUex** clef = 11011101011100 **OUex** 10101100101101 = 01110001110001

Etape5 :

De ce fait, le texte crypté obtenu en concaténant les résultats trouvés dans l'étape précédente est : **"000010110011000100000100001001110001110001"**

On se propose d'écrire un programme permettant de :

- saisir une chaîne binaire non vide représentant la clef de cryptage et ayant une longueur N multiple de 7 et inférieure ou égale à 21,
- crypter le contenu d'un fichier texte intitulé **"Initial.txt"**, ligne par ligne, et d'enregistrer le résultat dans un 2ème fichier texte intitulé **"Final.txt"** en appliquant le principe de cryptage décrit précédemment.

NB :

- Chaque ligne du fichier **"Initial.txt"** contient un texte dont la longueur ne dépasse pas 36 caractères.
- Le candidat n'est pas appelé à remplir le fichier **"Initial.txt"**.

Travail demandé :

1- Analyser le programme principal en le décomposant en modules.
2- Ecrire les algorithmes des modules envisagés.

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
```

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**

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 :

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

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

b) La formule $S = F_{n+2} - 1$ 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 π 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 ... $A_4A_3A_2A_1$"

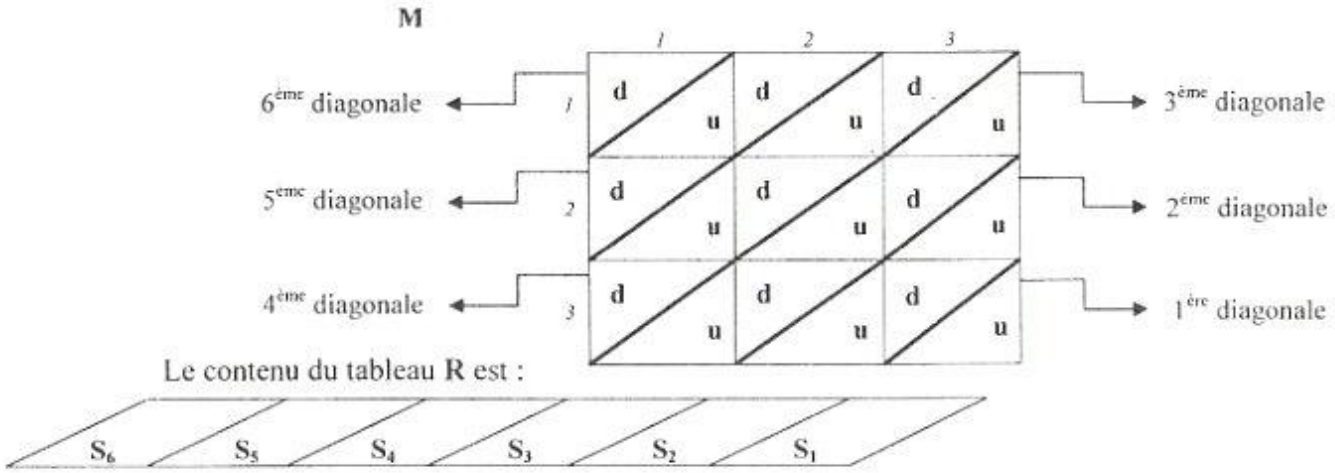
CB = "B_n ... $B_4B_3B_2B_1$"

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.



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 :

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

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 chaînes **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 1er chiffre de **CA** (7) par le 1er chiffre de **CB** (0) est 0

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

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

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

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.

Algorithmique et programmation
Session principale (2018)
Correction

N.B. :

- On n'acceptera pas toute solution sous forme d'analyse ou de traduction Pascal si on demande un algorithme.
- – **0.25** par erreur
- – **0.25** de la note attribuée à tous les TDO si la colonne Rôle est omise ou erronée.

Exercice 1 (3 points = 0.25 * 12)

1. L'opération de décalage est utilisée dans :

F le tri rapide

V le tri insertion

V le tri Shell

2. Le tri insertion est un cas particulier :

F du tri sélection

F du tri à bulle

V du tri Shell

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

F $P_0=0$; $P_n=3+P_{n-1}$

F $P_0=1$; $P_n=2*P_{n-1}$

V $P_0=1$; $P_n=3*P_{n-1}+1$

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

V

```
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
```

V

```
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
```

F

```
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
```

Ex n°2 (3 points)

a- L'algorithme de la fonction Fibo :

```
0) Def Fn Fibo(K:Entier) : Entier
1) Si K<2 Alors  Fibo ← 1
        Sinon Si k mod 2 = 0 Alors
```

Fibo ← Carré(Fibo(K div 2 - 1)) + Carré(Fibo(K div 2))
Sinon Fibo ← (2*Fibo(K div 2 +1) - Fibo(K div 2)) * Fibo(K div 2)
Fin Si

2) **Fin Fibo**

b- L'algorithme du module Fibo_Som :

0) **Def Fn Fibo_Som(n:Entier) : Entier**
1) Fibo_Som← Fn Fibo(n+2) – 1
2) **Fin Fibo_Som**

Le tableau de déclarations des objets locaux

Objet	Type/Nature	Rôle
Fibo	Fonction	Calculer un terme de la suite de Fibonacci

Exercice 3 (4 points)

1. Inconnue(5,2) = **Vrai**
   Inconnue(6,2) = **Faux**
   Inconnue(7,2) = **Vrai**
   Inconnue(9,2) = **Faux**

2. La fonction Inconnue **retourne la valeur Vrai si l'entier E** (avec E >2) **est premier, ou la valeur Faux dans le cas contraire**.
3. L'algorithme de la fonction **Calcul** :
   0) **Def Fn Calcul(Epsilon:Réel) : Rée1**
   1) R←4/3 , i←1
      Répéter
        i←i+2
        Si Fn Inconnue(i,2) Alors Ar←R
                                  R←R*Carré(i)/(Carré(i)-1)
        Fin Si
      Jusqu'à (Abs(RacineCarré(6*R) - RacineCarré(6*Ar))<Epsilon)
   2) Calcul←RacineCarré(6*R)
   3) **Fin Calcul**

Le tableau de déclarations des objets locaux

Objet	Type/Nature	Rôle
i	Entier	Compteur
R	Réel	Calculer $\pi^2/6$
Ar	Réel	Sauvegarder l'ancienne valeur de $\pi^2/6$
Inconnue	Fonction	Vérifier si i est premier

Problème (10 points)

1) Le tableau de déclarations des nouveaux types :

Type
Matrice = Tableau de 5 x 5 Octets
Tab = Tableau de 10 Octets

2) L'algorithme de la procédure **MiseAjour** :

0. Def Proc MiseAjour(Var R:Tab; N: Octet)

1. Pour i de 2*N à 2 (pas = -1) Faire

Si R[i] > 9 Alors R[i-1]←R[i-1] + R[i] Div 10

R[i] ← R[i] mod 10

Fin Si

Fin Pour

2. Fin MiseAjour

Le tableau de déclarations des objets locaux

Objet	Type/Nature	Rôle
I	Entier	Compteur

3) Analyse du programme principal :

❼Résultat = Ecrire(A, " * ", B, " = " , P)

❶A = Proc Saisie(A)

❷B = Proc Saisie(B)

❻P←Fn Result(R,2*Long(CA))

❺R= Proc Diag(M, Long(CA) , R)

❹M = Remplir(M,CA,CB)

❸(CA,CB) = [Convch(A, CA), Convch(B, CB)]

Tantque (Long(CA) ≠ long(CB)) Faire

Si Long(CA)< long(CB) Alors CA ← "0" + CA

Sinon CB ← "0" + CB

Fin Si

Fin Tantque

Le tableau de déclarations des objets globaux

Objet	Type/Nature	Rôle
A	Entier	Contenir l'entier A
B	Entier	Contenir l'entier B
CA	Chaine[5]	Une chaîne contenant les chiffres de l'entier A
CB	Chaine[5]	Une chaîne contenant les chiffres de l'entier B
P	Chaine[10]	Contenir l'équivalent du produit de A par B

M	Matrice	Contenir les résultats des produits des chiffres de A et de B
R	Tab	Contenir les sommes des chiffres des diagonales
Saisie	Procédure	Saisir un entier
Result	Fonction	Concaténer les chiffres du tableau R
Diag	Procédure	Remplir et mettre à jour le tableau R à partir des diagonales
Remplir	Procédure	Remplir M par les résultats des produits des chiffres de A et de B

4) Les algorithmes des modules envisagés :

a) L'algorithme de la procédure Saisie :

0) Def Proc Saisie(Var K:Entier)
1) Répéter
Ecrire("Saisir un entier ")
Lire(K)
Jusqu'à (K≥10) et (K≤10000)
2) Fin Saisie

b) L'algorithme de la procédure Remplir :

0) Def Proc Remplir(Var M:Matrice ; Cha,Chb:Chaîne)
1) Pour i de 1 à Long(Cha) Faire
Pour j de 1 à Long(Chb) Faire
M[i,j] ← (Ord(Cha[i]) - 48)*(Ord(Chb[j]) - 48)
Fin Pour
Fin Pour
2) Fin Remplir

Le tableau de déclarations des objets locaux

Objet	Type/Nature	Rôle
i	Octet	Compteur
j	Octet	Compteur

c) L'algorithme de la fonction Result

0) Def Fn Result (R:Tab ; K:Octet) : Chaîne
1) Res← ""
Pour i de K à 1 Faire
Res← Chr(R[i] + 48) + Res
Fin Pour
2) Result←Res
3) Fin Résultat

Le tableau de déclarations des objets locaux

Objet	Type/Nature	Rôle
i	Octet	Compteur
Res	Chaîne	Former le résultat du produit de A et B

Barème :

Exercice 1 : (3 points = 12 * 0.25)

On accepte les réponses V, F, Vrai, Faux

1. F-V-V **2. F-F-V** **3. F-F-V** **4. V-V-F**

Exercice n°2 : (3 points)

a) Fonction récursive Fibo (2.25 points)

Tâches	Points
Entête (Paramètres + type de la fonction)	0.5 (=0.25+0.25)
Condition + traitement d'arrêt	0.25+0.25
Test si n est pair	0.25
Appel récursif cas où n est pair	0.5
Appel récursif cas où n est impair	0.5

b) Fonction Fibo_som (0.75 points)

Tâches	Points
Entête (Paramètres + type de la fonction)	0.25
Appel de la fonction fibo et affectation	0.25
TDO	0.25

Exercice n°3 : (4 points)

1. Inconnue(5,2)=**Vrai** Inconnue(6,2)=**Faux** Inconnue(7,2)=**Vrai** Inconnue(9,2)=**Faux** **(1 point = 4 *0.25)**
2. **Vérifier** si un entier E est **premier : (0.5 point= 0.25+0.25)**
3. Calcul approché de π : Fonction calcul (epsilon) **: (2.5 points)**

Tâches	Points
Entête	0.25
Initialisations	0.25
Boucle + condition d'arrêt	0.5=0.25+0.25
Test premier avec la fonction Inconnue	0.25
Calcul du nouveau terme + affectation	0.5=0.25+0.25
Incrémentation du compteur	0.25
Affectation du résultat au nom de la fonction	0.25
TDO	0.25

Problème : (10 points)

1. TDNT **Matrice** et **Tab: (0.5 point = 0.25+0.25)**
2. Algorithme PROC MiseAjour (R,N) : **(1.50 point)**

Tâches	Points
Entête	0.25
Boucle	0.25
Test	0.25

Affectations de mise à jour	0.5
TDO	0.25

3. Analyse du Problème : (2 points)

Tâches	Points
➢ Modularité ➢ Cohérence (appels + conformité des paramètres)	0.5 1 = 0.5+0.5
➢ TDOG	0.5

4. Algorithmes des modules envisagés : (6 points)

Tâches	Points
➢ TDOL	1
➢ Saisie de A et de B : - Lecture - Contrainte	**1 =** 0.5 = 0.25 * 2 0.5 = 0.25 * 2
➢ **Conversion** de A et B en chaînes et **ajustement** des longueurs - Conversion de A et B en chaînes - Ajustement des longueurs (boucle + test + affectations)	**1.25 =** 0.5 = 0.25 * 2 1 = 0.25 +0.25+0.25
➢ Remplissage de la matrice (Boucles + affectation)	1 = 0.5 * 2
➢ **Appel** du module Diag et **détermination** du produit à partir de R - Appel du module Diag - Détermination du produit ▪ Initialisation ▪ Boucle ▪ Affectation	**1.25 =** 0.25 0.25 0.5 0.25
➢ Affichage du produit	0.5

RÉPUBLIQUE TUNISIENNE MINISTÈRE DE L'ÉDUCATION ●●●●● **EXAMEN DU BACCALAURÉAT** SESSION **2018**	**Session de contrôle**	
	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 données ci-dessous, mettre dans chaque case, la lettre **V** si la proposition est correcte ou la lettre **F** dans le cas contraire.

a) Un module est dit récursif, s'il comporte dans son corps :

- ☐ au plus un appel à lui même
- ☐ au moins deux appels à lui même
- ☐ un ou plusieurs appels à lui-même

b) Une fonction récursive doit comporter :

- ☐ un appel récursif en changeant au moins la valeur d'un paramètre
- ☐ une condition d'arrêt de l'appel récursif
- ☐ des variables locales

c) Lors de l'exécution d'un traitement récursif :

- ☐ le dernier appel doit être traité en premier
- ☐ le premier appel doit être traité en premier
- ☐ le dernier appel doit être traité en dernier

d) Un traitement récurrent dépend :

- ☐ toujours d'un seul traitement précédent
- ☐ de zéro traitement précédent
- ☐ d'un ou de plusieurs traitement(s) précédent(s)

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

Important :

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

Objet	*Type/Nature*	*Rôle*

Exercice 2 : (4,5 points)

Soient les deux fonctions **f** et **g** définies comme suit :

- $f(x) = x \quad avec\, x \in \mathbb{R}$
- $g(x) = \cos(x) \quad avec\, x \in \mathbb{R}$

1- Ecrire un algorithme d'une fonction **Calcul (epsilon)** permettant de calculer une valeur approchée, à epsilon près, de **p** tel que **cos(p) = p**.

2- Soit le graphique suivant représentant les courbes des deux fonctions f et g et de la droite $x = p$.

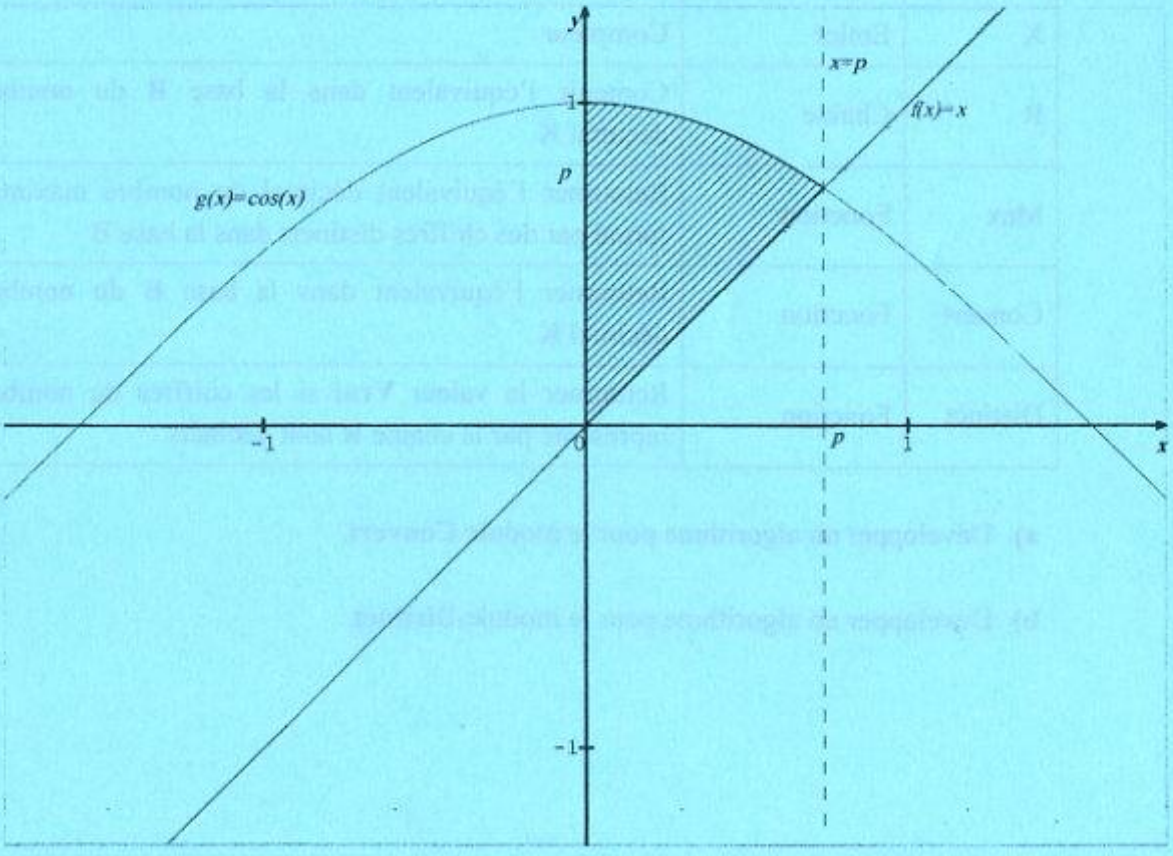


Figure 1

Ecrire un algorithme d'une fonction **Surface (epsilon)** qui permet de calculer une valeur approchée, à epsilon près, de l'aire délimitée par les deux courbes des deux fonctions **f** et **g**, l'axe des ordonnés et la droite **x = p** (l'aire hachurée dans ***Figure 1***).

Exercice 3 : (4 points)

On se propose de trier un tableau **T** de **n** entiers (**avec n dans [5..20]**), en utilisant le **tri par insertion dichotomique** qui repose sur le même principe du tri par insertion, mais utilise la recherche dichotomique pour déterminer la position d'insertion.

On rappelle que le principe de tri par insertion consiste à rechercher, séquentiellement pour chaque élément d'un tableau, sa position d'insertion dans la portion du tableau qui le précède, de décaler si c'est nécessaire et de l'insérer pour que cette portion du tableau reste triée et ce en commençant de l'élément numéro **2** jusqu'au dernier.

1- Soit l'algorithme incomplet de la fonction récursive **Dichotomie** ci-dessous permettant de retourner la position d'insertion d'un entier **k** dans la portion triée dans l'ordre croissant et délimitée par les indices **g** et **d** d'un tableau **T** d'entiers :

```
0) DEF FN Dichotomie (T : Tab ; g, d, k : Octet) : Octet
1) ..........
   Si g>d  Alors  Dichotomie ← g
   Sinon Si T[mil] = k  Alors  Dichotomie ← mil
   Sinon Si T[mil] > k  Alors ..........
   Sinon Si T[mil] < k  Alors ..........
   FinSi
2) Fin Dichotomie
```

a) Réécrire l'algorithme de la fonction **Dichotomie** et le compléter en plaçant les instructions ci-après aux bons endroits.

- *Dichotomie ← Dichotomie (T, g, mil-1, k)*
- *mil ← (g+d) DIV 2*
- *Dichotomie ← Dichotomie (T, mil+1, d, k)*

b) Dresser le tableau de déclaration du nouveau type **Tab**.

c) Dresser le tableau de déclaration des variables locales de la fonction **Dichotomie**.

2- En utilisant la fonction **Dichotomie** définie précédemment, écrire un algorithme d'un module qui permet de trier un tableau **T** de **n** entiers dans l'ordre croissant en appliquant la méthode de **tri par insertion dichotomique**.

Exercice 4 : (4 points)

Soit un circuit série formé par un générateur de tension constante, un interrupteur **K**, une bobine d'inductance **L** et de résistance interne **r** et un résistor de résistance R_0.

Afin de vérifier l'expression $i = I_0\left(1 - e^{-\frac{t}{\tau}}\right)$ qui détermine la valeur de l'intensité **i** à un instant **t**, un expérimentateur a stocké dans un fichier d'enregistrements **F_intens.dat** situé sur la racine du disque **C**, les mesures effectuées de l'intensité du courant **i** à plusieurs instants **t**. Chaque enregistrement est formé de deux champs, le premier comporte la valeur de **t** et le deuxième comporte la valeur de **i**.

On se propose d'écrire un programme qui permet de vérifier le degré de réussite de l'expérience et ce en comparant les valeurs expérimentales de **i** avec celles calculées théoriquement en utilisant la formule précédente. L'expérience sera dite réussie si la différence entre la valeur expérimentale et la valeur théorique ne dépasse pas 10^{-3} dans **90%** des mesures.

Travail demandé :

1- Ecrire une instruction d'assignation du fichier **F_intens.dat** à une variable logique **F**.

2- Donner une déclaration pour le type du fichier **F** ainsi que pour tout nouveau type nécessaire à sa déclaration.

3- Ecrire l'algorithme d'un module qui permet de remplir le fichier **F_intens.dat** par les valeurs expérimentales. La saisie se termine en répondant par "**N**" à la question " **Voulez vous saisir les valeurs d'une expérience (O/N) ?** ".

4- Ecrire l'algorithme d'un module qui permet de vérifier si l'expérience est réussie ou non.

On donne E=6V, L=300mH, r=10Ω et R0=140Ω avec R=R0 + r, $\tau = \frac{L}{R}$ et $I_0 = \frac{E}{R}$

NB :

- En remplaçant **E**, **L**, R_0 et **r** par leurs valeurs, l'expression de l'intensité devient :

  $i = \left(1 - e^{-\frac{t}{2}}\right)/25$.

- Pour calculer l'exponentiel e^x, on utilise la fonction prédéfinie **exp(x)**.

Exercice 5 : (4.5 points)

Dans une base **B**, un nombre est dit distinct s'il est composé par des chiffres distincts.

Exemple :

Dans la base **B = 2**, il y a trois nombres distincts qui sont : **0**, **1** et **10**

1- Donner tous les nombres distincts dans la base **3**.

2- On présente ci-dessous l'algorithme d'une procédure **Nbre_Distincts** qui permet d'afficher tous les nombres distincts d'une base **B**.

```
0) Def Proc Nbre_Distincts (B : Octet)
1) Pour K de 0 à Fn Max(B) Faire
     R ← Fn Convert(K, B)
     Si Fn Distinct(R) Alors Ecrire(R)
     Fin Si
   Fin Pour
2) Fin Nbre_Distincts
```

Tableau de déclaration des objets locaux

Objet	Type/Nature	Rôle
K	Entier	Compteur
R	Chaîne	Contenir l'équivalent dans la base **B** du nombre décimal **K**
Max	Fonction	Retourner l'équivalent décimal du nombre maximal formé par des chiffres distincts dans la base B
Convert	Fonction	Retourner l'équivalent dans la base **B** du nombre décimal **K**
Distinct	Fonction	Retourner la valeur **Vrai** si les chiffres du nombre représenté par la chaîne **R** sont distincts

a) Développer un algorithme pour le module **Convert**.

b) Développer un algorithme pour le module **Distinct**.

Algorithmique et programmation
Session de contrôle
Correction

Exercice 1 (3 points = 0,25 * 12)

a) Un module est dit récursif, s'il comporte dans son corps :

- **F** au plus un appel à lui même
- **F** au moins deux appels à lui même
- **V** un ou plusieurs appels à lui-même

b) Une fonction récursive doit comporter :

- **V** un appel récursif en changeant au moins la valeur d'un paramètre
- **V** une condition d'arrêt de l'appel récursif
- **F** des variables locales

c) Lors de l'exécution d'un traitement récursif :

- **V** le dernier appel doit être traité en premier
- **F** le premier appel doit être traité en premier
- **F** le dernier appel doit être traité en dernier

d) Un traitement récurrent dépend :

- **F** toujours d'un seul traitement précédent
- **F** de zéro traitement précédent
- **V** d'un ou de plusieurs traitement(s) précédent(s)

Exercice 2 (4,5 points)

1. **L'algorithme de la fonction Calcul** :

0) **Def Fn Calcul (epsilon:Réel) : Réel**
1) X←0
Répéter
Xprec←X
X←Cos(Xprec)
Jusqu'à (Abs(X-Xprec) ≤ epsilon)
2) Calcul←X
3) **Fin Calcul**

Le tableau de déclarations des objets locaux

Objet	Type/Nature	Rôle
X	Réel	Contenir le cosinus d'un réel
Xprec	Réel	Contenir la valeur précédente de X

2. L'algorithme de la fonction Surface :

0) Def Fn Surface (epsilon:Réel) : Réel

1) n←1 , P←Fn Calcul(epsilon), S←Fn Sur(0,p,n)
Répéter
n←n+1
As←S
S←Fn Sur(0,P,n)
Jusqu'à (Abs(S-As) ≤ epsilon)

2) Surface ←S

3) Fin Surface

Le tableau de déclarations des objets locaux

Objet	Type/Nature	Rôle
n	Entier	Le nombre des subdivisions
S	Réel	Contenir la valeur de la surface
As	Réel	Sauvegarder la valeur précédente de la surface
P	Réel	Contenir une valeur approchée à epsilon prés de P tel que cos(P)=P
Calcul	Fonction	Déterminer une valeur approchée à epsilon prés de P tel que cos(P)=P
Sur	Fonction	Calculer une valeur approchée de la surface

L'algorithme de la fonction Sur

0) Def Fn Sur (a,b:Réel ; n:Entier) : Réel

1) h←(b-a)/n, x←a, s←(cos(a)-a + cos(b)-b)/2
Pour i de 1 à n-1 Faire
x←x+h
s←s+cos(x)-x
Fin Pour

2) Sur←h*s

3) Fin Sur

Le tableau de déclarations des objets locaux

Objet	Type/Nature	Rôle
h	Réel	Contenir la valeur du diamètre
x	Réel	Contenir la valeur de l'abscisse
s	Réel	Contenir la somme des ordonnées
i	Entier	Compteur

Exercice 3 (4 points)

1)-a) L'algorithme de la fonction dichotomie :

0) Def Fn Dichotomie (T:Tab ; g,d,k:Octet) : Octet

1) Mil ← (g+d) DIV 2
Si (g>d) Alors Dichotomie← g
Sinon Si (T[mil]=k) Alors Dichotomie← mil
Sinon Si (T[mil]>k) Alors **Dichotomie ←Dichotomie (T, g, mil-1, k)**
Sinon Si (T[mil]< k) Alors **Dichotomie ←Dichotomie (T, mil+1, d, k)**
FinSi

2) Fin Dichotomie

1)-b) Le tableau de déclarations du nouveau type :

Type
Tab= Tableau de 20 Entiers

1)-c) Le tableau de déclarations des objets locaux :

Objet	Type/Nature	Rôle
Mil	Octet	Contenir l'indice du milieu

2)

0) Def Proc TriInsertionDichotomique (Var T:Tab ; N:Octet)

1) Pour i de 2 à N Faire

X←T[i]

P←Fn Dichotomie(T,1,i-1,X)

Si (P< i) Alors

Proc Decalage(T,P,i-1)

T[P]←X

Fin Si

Fin Pour

2) Fin TriInsertionDichotomie

Le tableau de déclarations des objets locaux

Objet	Type/Nature	Rôle
i	Octet	Compteur
X	Octet	Sauvegarder la valeur de l'élément d'indice i
P	Octet	Contenir l'indice de la position d'insertion
Dichotomie	Fonction	Rechercher la position d'insertion
Decalage	Procédure	Décaler des éléments du vecteur T

L'algorithme de la procédure Decalage

0) Def Proc Decalage (Var T:Tab;deb,fin: Octet)

1) Pour j de fin à deb (pas= - 1) Faire

T[j+1] ←T[j]

Fin Pour

2) Fin Decalage

Le tableau de déclarations des objets locaux

Objet	Type/Nature	Rôle
j	Octet	Compteur

Exercice 4 (4 points)

1) L'association au fichier "F_intens.dat" : **Associer (F, "C:\F_intens.dat")**

2) Le tableau de déclarations du nouveau type :

Type
Mesure = Enregistrement **Temps : Entier long** **Intensite : Réel** **Fin Mesure**
Valeurs = Fichier de Mesure

***Nb** : pour le champ **Temps** on acceptera tout type numérique.*

3) L'algorithme de la procédure Remplir :

0) Def Proc Remplir (Var F:Valeurs)

1) Recréer(F)
   Répéter
      Ecrire ("Saisir le temps "), Lire(E.Temps)
      Ecrire ("Saisir l'intensité relatif au temps saisi "), Lire(E.Intensite)
      Ecrire(F, E)
      Répéter
            Ecrire ("Voulez-vous saisir les valeurs d'une expérience (O/N) ? ")
            Lire(Rep)
      Jusqu'à Rep dans["O","N"]
   Jusqu'à Rep = "N"
2) Fermer(F)
3) **Fin Remplir**

Le tableau de déclarations des objets locaux

Objet	**Type/Nature**	**Rôle**
E	Mesure	Contenir le temps et l'intensité d'une mesure
Rep	Caractère	Contenir la réponse de l'utilisateur

4) L'algorithme de la fonction Verifier :

0) Def Fn Verifier (Var F:Valeurs) : Booléen

1) Ouvrir(F), Nbr←0
   Tantque Non(Fin_Fichier(F)) Faire
      Lire(F, E)
      T ←E.Temps
      MesPrat←E.Intensite
      MesTh←(1-Exp(-T/2))/25
      Si Abs(MesPrat - MesTh)<0.001 Alors Nbr←Nbr+1
      FinSi
   Fin Tantque
2) Verifier ← Nbr>(90*Taille_Fichier(F)/100)
3) Fermer(F)
4) **Fin Verifier**

Le tableau de déclarations des objets locaux

Objet	**Type/Nature**	**Rôle**
E	Mesure	Contenir le temps et l'intensité d'une mesure
Nbr	Entier Long	Contenir le nombre des expériences dont la différence entre la valeur expérimentale et la valeur théorique ne dépasse pas 10^{-3}
T	Entier Long	Contenir le temps d'une mesure
MesPrat	Réel	Contenir la mesure pratique
MesTh	Réel	Contenir la mesure théorique

Exercice 5 (4.5 points)

1) Les nombres distincts dans la base 3 sont : **0**, **1**, **2**, **10**, **12**, **20**, **21**, **102**, **120**, **201** et **210**

2) Développement des algorithmes des modules :

a. La fonction Convert

0) Def Fn Convert (D:Entier ; B:Octet) : Chaîne

1) R← ""

Répéter

Reste ← D mod B

Si Reste<10 Alors R←Chr(48 + Reste) + R

Sinon R←Chr(55 + Reste) + R

Fin Si

D←D div B

Jusqu'à D = 0

2) Convert←R

3) Fin Convert

Le tableau de déclarations des objets locaux

Objet	Type/Nature	Rôle
R	Chaîne	Contenir l'équivalent dans la base B du nombre décimal D
Reste	Octet	Contenir le reste de la division entière par B

b. La fonction Distinct

0) Def Fn Distinct (R : Chaine) : Booléen

1) Si Long[R] = 1 Alors Distinct ← Vrai

Sinon Si Pos(R[1], SousChaine(R,2,Long(R) -1))>0 Alors Distinct ← Faux

Sinon Distinct ← Fn Distinct(SousChaine(R,2,Long(R)-1))

Fin Si

2) Fin Distinct

Barème

N.B. :

- On acceptera toute autre solution correcte.
- On n'accepte que les solutions sous forme d'algorithme.
- – **0.25** par erreur
- – **0.25** de la note attribuée au TDO si la colonne Rôle est omise ou erronée.

Exercice n°1 : (3 points = 12 * 0.25)

On accepte les réponses V, F, Vrai, Faux

1. F-F-V **2. V-V-F** **3. V-F-F** **4. F-F-V**

Exercice n°2 : (4.5 points)

a) Fonction calcul (epsilon) : (2 points)

Tâches	Points
Entête	0.25
Initialisation	0.25
Boucle + condition d'arrêt	0.5= 0.25+0.25
Modification de la valeur de x	0.5
Affectation du résultat au nom de la fonction	0.25
TDO	0.25

b) Fonction Surface (epsilon) (2.5 points)

Tâches	Points
Entête	0.25
Appel de la fonction calcul	0.25
Boucle + condition d'arrêt	0.5
Incrémentation du nombre d'intervalles	0.25
Calcul de la surface (initialisations + boucle + affectations)	0.75 = 0.25 * 3
Affectation du résultat au nom de la fonction	0.25
TDO	0.25

Exercice n°3 : (4 points)

a) Placement des instructions aux bons endroits : **(0.75 point = 0.25 *3)**

b) TDNT **Tab : (0.25 point)**

c) TDOL : (**0.25 point)**

d) Module tri par insertion dichotomique **: (2.75 points)**

Tâches	Points
Entête	0.25
Boucle	0.25
Sauvegarde de T[i]	0.25
Recherche de la position d'insertion (Appel de la fonction Dichotomie + paramètres)	0.5= 0.25+0.25
Décalage (boucle + affectation)	0.75 = 0.5+ 0.25
Affectation d'insertion de T[i]	0.25
TDO	0.5

Exercice n°4 : (4 points)

1. Association : **(0.25 point)**
2. TDNT **(enregistrement + fichier) : (0.5 point= 0.25+0.25)**
3. Remplissage du fichier F_intensite.dat **: (1.25 points)**

Tâches	Points
Création du fichier + Fermeture du fichier	0.25
Boucle + condition d'arrêt	0.25
Lecture du temps + Lecture de l'intensité	0.25
Ecriture dans le fichier	0.25
Lecture de la réponse (O/N)	0.25

4. Vérification de la réussite de l'expérience **: (1.75points)**

Tâches	Points
Ouverture du fichier + Fermeture du fichier	0.25
Initialisation du nombre d'expériences réussies	0.25
Parcours du fichier	0.25
Lecture des valeurs expérimentales : temps + intensité	0.25
Calcul théorique de l'intensité	0.25
Comparaison + incrémentation du nombre d'expériences réussies	0.25
Affectation du résultat de vérification du degré de réussite	0.25

NB : Les Entêtes +les TDO des deux questions 3°/ et 4°/ : **0.25 point**

Exercice n°5 : (4.5 points)

1. Nombres distincts dans la base 3 : **(1 point)**
2. Algorithme du module **Convert** : **(2.25 points)**

Tâches	Points
Entête	0.25
Initialisation	0.25
Boucle + condition d'arrêt	0.25
Calcul du reste	0.25
Test par rapport à 10	0.25
Affectation cas reste < 10	0.25
Affectation cas reste >= 10	0.25
Calcul du quotient	0.25
TDO	0.25

3. Algorithme du module **Distinct** : **(1.25 points)**

Tâches	Points
Entête	0.25
Parcours de la chaîne	0.25
Vérification de l'unicité de chaque caractère	0.50
Affectation du résultat au nom de la fonction	0.25

RÉPUBLIQUE TUNISIENNE MINISTÈRE DE L'ÉDUCATION ●●○●● **EXAMEN DU BACCALAURÉAT** **SESSION 2017**	*Épreuve :* **Algorithmique et Programmation** *Section :* **Sciences de l'informatique** *Durée :* **3h** — Coefficient : **2.25** ***Session principale***

Exercice 1 : (2,25 Points)

Soit l'algorithme ci-dessous de la fonction **Rectangle** permettant de calculer, en utilisant la méthode des rectangles, l'aire résultante de la courbe de la fonction $f(x) = \frac{6}{1+x}$ sur un intervalle **[a,b]** subdivisé en **n** rectangles.

```
0) DEF FN Rectangle (a, b : Réel ; n : Entier) : Réel
1) h ← (b – a)/n
   S ← 0
   x ← a
   Pour i de 1 à n Faire
         S ← S + 6/(1+x)
         x ← x + h
   Fin Pour
2) Rectangle ← S * h
3) FIN Rectangle
```

Travail demandé :

Pour chacune des questions suivantes, valider chaque proposition par **V** si la réponse est correcte ou par **F** dans le cas contraire.

1) La fonction **Rectangle** permet de calculer l'aire résultante de la courbe de la fonction **f** sur un intervalle [a,b] selon la méthode des :

☐	**rectangles à gauche**
☐	**rectangles du point milieu**
☐	**rectangles à droite**

2) Pour les valeurs **a = 1**, **b = 5** et **n = 4**, le résultat retourné par la fonction **Rectangle** est :

☐	**5.5**
☐	**7.7**
☐	**10.12**

3) Pour appliquer la méthode des trapèzes au lieu de la méthode des rectangles, on remplace l'instruction de calcul de la somme **S** par :

☐	**S ← S + (6/(1+x) + 6/(1+x+h))/2**
☐	**S ← S + 6/(1+x+h)/ 2**
☐	**S ← S + (6/(1+x) - 6/(1+x+h))/2**

Exercice 2 : (2,75 points)

Soit x un réel de l'intervalle **]0, 1[** .

En binaire, x s'écrit sur **n** chiffres après la virgule comme suit : $\mathbf{0.c_1c_2c_3c_4c_5\ldots c_{n-1}c_n}$ avec $\mathbf{c_i}$ un chiffre binaire (0 ou 1).

Pour déterminer les chiffres $\mathbf{c_i}$ après la virgule de l'équivalent binaire du réel x, on suit le procédé suivant :

1. calculer $\mathbf{c_1}$ en multipliant x par **2**,
   - si $2 * x < 1$, alors $\mathbf{c_1}$ est égal à **zéro** et on remplace x par $2 * x$
   - si $2 * x \geq 1$, alors $\mathbf{c_1}$ est égal à **1** et on remplace x par $2 * x - 1$
2. répéter **n** fois l'étape **1** jusqu'à calculer $\mathbf{c_n}$.

Exemple 1 :

Pour $x = 0.825$ et $n = 5$, l'équivalent binaire de x est $0.c_1c_2c_3c_4c_5$ et se calcule comme suit :

- $2 * 0.825 = 1.65$ d'où $c_1 = 1$ et on remplace x par 0.65 (1.65 – 1)
- $2 * 0.65 = 1.3$ d'où $c_2 = 1$ et on remplace x par 0.3 (1.3 – 1)
- $2 * 0.3 = 0.6$ d'où $c_3 = 0$ et on remplace x par 0.6 (2*0.3)
- $2 * 0.6 = 1.2$ d'où $c_4 = 1$ et on remplace x par 0.2 (1.2 – 1)
- $2 * 0.2 = 0.4$ d'où $c_5 = 0$

D'où l'équivalent binaire à **5** chiffres après la virgule de **0.825** est **0.11010**

Exemple 2 :

Pour $x = 0.625$ et $n = 4$, l'équivalent binaire de x est $0.c_1c_2c_3c_4$ et se calcule comme suit :

- $2 * 0.625 = 1.25$ d'où $c_1 = 1$ et on remplace x par 0.25 (1.25 -1)
- $2 * 0.25 = 0.5$ d'où $c_2 = 0$ et on remplace x par 0.5 (0.25*2)
- $2 * 0.5 = 1.0$ d'où $c_3 = 1$ et on remplace x par 0 (1.0 - 1)
- $2 * 0 = 0$ d'où $c_4 = 0$

D'où l'équivalent binaire à **4** chiffres après la virgule de **0.625** est **0.1010**

Travail demandé :

Ecrire un algorithme d'une fonction qui retourne la représentation binaire, sur **n** chiffres après la virgule, d'un réel x de l'intervalle **]0, 1[** .

NB :

- x et **n** sont passés en paramètres et ils sont déjà saisis dans le module appelant.
- Chaque algorithme proposé doit être accompagné d'un tableau de déclaration des objets ayant la forme suivante :

Objet	Type / Nature	Rôle

Exercice 3 : (5 points)

Pour **A** et **B** deux entiers strictement positifs, on définit la relation suivante :

Si **(A! * B! mod (A+B) = A) OU (A! * B! mod (A+B) = B)**

alors **(A+B) est un nombre premier**

(Avec A! et B! sont respectivement la factorielle de A et celle de B)

Exemples :

A	B	A+B	A!	B!	A! * B!	A! * B! mod (A+B)	A+B
2	3	5	2	6	12	12 mod 5 = 2 (= A)	2+3 = 5 est premier
7	4	11	5040	24	120960	120960 mod 11 = 4 (= B)	7+4 = 11 est premier
5	2	7	120	2	240	240 mod 7 = 2 (= B)	5+2 = 7 est premier

Travail demandé :

En disposant d'un fichier texte nommé "**Source.txt**" contenant dans chaque ligne un couple de deux valeurs séparées par un espace représentant respectivement les valeurs de deux entiers **A** et **B**, écrire un algorithme d'un module, qui à partir du fichier "**Source.txt**", permet :

1. de générer un nouveau fichier texte "**Resultat.txt**" contenant dans chaque ligne, les valeurs du couple **A** et **B**, vérifiant la relation définie précédemment, sous la forme d'un nombre complexe comme suit : "**A+i*B**"
2. d'afficher le contenu du fichier "**Resultat.txt**"

Exemple :

Pour le contenu du fichier "**Source.txt**" suivant :

```
2 3
4 5
7 4
6 3
5 2
```

Le fichier "**Resultat.txt**" aura le contenu suivant :

```
2+i*3
7+i*4
5+i*2
```

Problème : (10 points)

On se propose de réaliser un moteur de recherche local permettant de trouver, sur un ordinateur, tous les fichiers textes contenant un ensemble de mots saisis par l'utilisateur.

Pour cela, on dispose d'un fichier texte nommé "**Chemin.txt**" situé sur la racine du disque C et contenant les chemins d'accès des fichiers textes du disque local de l'ordinateur à raison d'un chemin par ligne, sachant que :

- un chemin d'accès est composé d'au maximum **80** caractères
- le nombre maximum de fichiers texte est égal à **100**

Le procédé de recherche consiste à :

- saisir dans un tableau **TM** les **N** mots (**0 < N < 11**) à rechercher. Un mot est formé uniquement par des lettres,
- remplir une matrice **M** par le nombre de lignes, contenant le mot à rechercher, dans chaque fichier tels que :

  M[i,j] = nombre de lignes du fichier **G** contenant le mot à rechercher **TM[j]**

  Où **G** représente le fichier dont son chemin est indiqué dans la ligne numéro **i** du fichier **"Chemin.txt"**
- afficher tous les mots à rechercher suivis par les chemins des fichiers qui les contiennent s'ils existent séparés par un espace.

Exemple :

Pour :

- le fichier "**Chemin.txt**" suivant :

C:\bac2017\matieres.txt
C:\bac2017\programs.txt
C:\divers\textes\web.txt
C:\revision.txt
C:\Application\Exercice.txt

- **N = 4** et le tableau **TM** suivant :

Informatique	*Algorithme*	*Html*	*Php*

Si la recherche des nombres d'occurrences des mots clés donne la matrice **M** suivante :

	1	*2*	*3*	*4*
1	0	1	0	0
2	1	0	0	0
3	0	0	0	4
4	0	0	0	0
5	0	0	0	10

Alors le programme Affichera :

Informatique : *C:\bac2017\Programs.txt*
Algorithme : *C:\bac2017\Matieres.txt*
Html :
Php: *C:\divers\textes\Web.txt* *C:\Application\Exercice.txt*

Travail demandé :

1- Analyser le problème en le décomposant en modules.
2- Ecrire un algorithme solution pour chaque module envisagé. Chaque algorithme proposé doit être accompagné d'un tableau de déclaration des objets ayant la forme suivante :

Objet	Type / Nature	Rôle

Corrigé *: Algorithmique et Programmation*

Section : Sciences de l'informatique

Session principale Baccalauréat 2017

Exercice N°1 : (2.25 points = 9*0.25)

Pour chacune des questions suivantes, valider chaque proposition par **V** si la réponse est correcte ou par **F** dans le cas contraire.

1) La fonction **Rectangle** permet de calculer la valeur de l'aire par la fonction **f** selon la méthode de :

V	**rectangles à gauche**
F	**rectangles du point milieu**
F	**rectangles à droite**

2) Pour les valeurs **a = 1**, **b = 5** et **n = 4**, le résultat retourné par la fonction **Rectangle** est :

F	**5.5**
V	**7.7**
F	**10.12**

3) Pour appliquer la méthode de trapèzes au lieu de la méthode de rectangles, on remplace l'instruction de calcul de la somme **S** par :

V	**S ← S + (6/(1+*x*) + 6/(1+*x*+h))/2**
F	**S ← S + 6/(1+*x*+h)/ 2**
F	**S ← S + (6/(1+*x*) - 6/(1+*x*+h))/2**

Exercice N°2 : (2.75 points)

0) Def Fn Nom (x : Réel ; n : octet) : chaîne
1) R←"0."
Pour i de 1 à n Faire
x← 2 * x
Si x < 1 Alors R ← R + "0"
Sinon R ← R + "1"
x ← x – 1
Fin Si
Fin Pour
2) Nom ← R
3) Fin Nom

Tableau des déclarations des objets locaux

Objet	Type / Nature	Rôle
R	Chaine de caractères	Contenir l'équivalent binaire du réel x
i	Octet	Compteur

Exercice N°3 : (5 points)

0) DEF PROC Exercice3 (Var FSource, FResult:Text)
1) Associer (FSource, "C:\Source.txt"), Associer (FResult, "C:\Result.txt"), Ouvrir (FSource), Recréer (FResult)

```
Tant que Non (Fin_Fichier (FSource)) Faire
                    Lire_nl (FSource, A, B)
                    Si (FN Fact(A) * FN Fact(B) mod (A+B) dans [A, B])
                    Alors Écrire_nl (FResult, A, "+i*", B)
                          Écrire (A, "+i*", B)
                    Fin Si
  Fin Tant que
```

2) Fermer (FSource), Fermer (FResult)

3) Fin Exercice3

Objet	Type / Nature	Rôle
A	Entier	La première valeur
B	Entier	La deuxième valeur
Fact	Fonction	Calcul de la factorielle d'un entier

0) DEF FN Fact (N:Entier) : EntierLong

1) F ← 1

```
Pour i de 1 à N Faire
    F ← F * i
Fin Pour
```

2) Fact ← F

3) Fin Fact

Objet	Type / Nature	Rôle
i	Entier	Compteur
F	EntierLong	Sauvegarde la factorielle

Problème : (10 points)

Analyse du programme principal :

Résultat = Proc Afficher (M, TM, N, nl, F)
N, TM = Proc Remplir (TM, N)
M, nl, F = Associer (F,"Chemin.txt"),
Proc FormerMatrice(M, TM, N, nl, F)

Tableau de déclaration des nouveaux types

Type
Mot = tableau de 10 Chaînes de caractères
Matrice = tableau de 100 x 10 Octets

Tableau de déclaration des objets globaux

Objet	Type / Nature	Rôle
TM	Mot	Contenir les mots à chercher
M	Matrice	Contenir la fréquence de chaque mot
F	Texte	Fichier contenant des chemins de fichiers textes
N	Octet	Nombre de mots du tableau TM
nl	Octet	Nombre de lignes du fichier F
FormerMatrice	Procédure	Remplir la matrice M
Afficher	Procédure	Afficher les mots suivis par les chemins qui les contiennent
Remplir	Procédure	Saisir N et remplir le tableau TM par N mots

Algorithme de la Procédure Afficher :

⓿ Def Proc Afficher (M : Matrice ; TM : Mot ; N, nl : Octet ; Var F : Texte)
❶ Pour j de 1 à N Faire
    Ecrire (TM[j]," : ")
    Ouvrir(F)
    Pour i de 1 à nl Faire
        Lire_nl(F, ch)
        Si M[i,j] > 0 Alors écrire(ch, " ")
        Fin Si
    Fin Pour
  Fin Pour
❷ Fin Affichage

Tableau de déclaration des objets locaux

Objet	Type / Nature	Rôle
i, j	Octet	Compteur
ch	Chaîne de caractères	Contenu de la ligne i du fichier F

Algorithme de la procédure Remplir :

⓿ Def Proc Remplir (Var TM : Mot ; Var N : Octet)
❶ Répéter
    Ecrire ("Donner le nombre de mots à chercher : ")
    Lire(N)
  Jusqu'à N dans [1.. 10]
❷Pour i de 1 à N Faire
    Répéter
        Ecrire ("Donner le mot n° ",i," : ")
        Lire(TM[i])
    Jusqu'à (Long(TM[i])>0) et (Fn Alpha(TM[i])=vrai)
  Fin Pour
❸Fin Remplir

Tableau de déclaration des objets locaux

Objet	Type / Nature	Rôle
i	Octet	Compteur
Alpha	Fonction	Vérifier qu'une chaîne est formée uniquement par des lettres

Algorithme de la fonction Alpha :

⓿ Def FN Alpha (ch : Chaîne) : booléen
❶ i←0
  Répéter
    i←i+1
  Jusqu'à (NON (Majus (ch[i]) Dans ["A".."Z"])) ou (i=Long (ch))
❷ Alpha←Majus (ch[i]) Dans ["A".."Z"]
❸ Fin Alpha

Tableau de déclaration des objets locaux

Objet	Type / Nature	Rôle
i	Octet	Compteur

Algorithme de la procédure FormerMatrice

```
❶ Def Proc FormerMatrice (Var M : Matrice ; TM : Mot ; N : Octet ; Var nl : Octet ; Var F :Texte)
❷ Ouvrir(F), nl← 0
   Tant que Non (Fin_Fichier(F)) Faire
                Lire_nl (F, Ch)
                Associer (G, Ch)
                nl← nl+1
                Pour c de 1 à N Faire
                        Ouvrir(G)
                        nb ← 0
                        Tant que Non (Fin_Fichier(G)) Faire
                                Lire_nl (G, Lig)
                                Si Pos (TM[c], lig) > 0 Alors nb←nb+1
                                Fin Si
                        Fin Tant que
                        M[nl, c] ← nb
                Fin Pour
                Fermer(G)
        Fin Tant que
        Fermer(F)
❸Fin FormerMatrice
```

Tableau de déclaration des objets locaux

Objet	Type / Nature	Rôle
c	Octet	Compteur de colonne
Ch	Chaîne de caractères	Stocker un chemin d'un fichier
G	Texte	Fichier texte correspondant au chemin Ch
Lig	Chaîne de caractères	Le contenu d'une ligne du fichier F
nb	Octet	Le nombre de ligne du fichier G contenant le mot n° C du tableau TM

RÉPUBLIQUE TUNISIENNE MINISTÈRE DE L'ÉDUCATION ●●○●● **EXAMEN DU BACCALAURÉAT** **SESSION 2017**	*Épreuve :* **Algorithmique et Programmation**	
	Section : **Sciences de l'informatique**	
	Durée : **3h**	Coefficient : **2.25**
	Session de contrôle	

Le sujet comporte 04 pages

Exercice 1 : (3 points)

Soit le tableau de déclaration des nouveaux types suivant :

Type
Personne = Enregistrement id, age : entier genre : caractère Fin Personne
Tab1 = tableau de 100 **Personne** **Tab2** = tableau de 200 **Personne**

Soient **H** et **F** deux tableaux contenant respectivement **n1** et **n2** enregistrements de type **Personne** et triés selon l'ordre croissant du champ **id**.

Et soit l'algorithme de la procédure **Traitement** suivant :

```
0) Def Proc Traitement (n1,n2 : Octet ; H, F : Tab1 ; Var P : Tab2)
1) k←0, i ← 0, j←0
   Répéter
        k←k+1
        i←i+1
        P[k] ← H[i]
        k←k+1
        j←j+1
        P[k] ← F[j]
   Jusqu'à (i=n1) ou (j=n2)
   Si (i=n1) Alors
            Pour c de j+1 à n2 Faire
                 k←k+1
                 P[k] ← F[c]
            FinPour
       Sinon
            Pour c de i+1 à n1 Faire
                 k←k+1
                 P[k] ← H[c]
            FinPour
   FinSi
2) Fin Traitement
```

Travail demandé :

1) Donner le contenu du tableau **P** après exécution de la procédure **Traitement** pour **n1=4, n2=6** et les valeurs de **H** et **F** suivantes :

H	id = 123 age = 57 genre = "M"	id = 125 age = 22 genre = "M"	id = 363 age = 35 genre = "M"	id = 430 age = 33 genre = "M"		
F	id = 113 age = 57 genre = "F"	id = 115 age = 30 genre = "F"	id = 263 age = 18 genre = "F"	id = 380 age = 55 genre = "F"	id = 455 age = 23 genre = "F"	id = 663 age = 19 genre = "F"

2) Déduire le rôle de la procédure **Traitement**.
3) Apporter les modifications nécessaires au contenu de la boucle **Répéter** pour obtenir le tableau **P** trié selon le champ **id**.

Exercice 2 : (3 points)

Soient **a** et **n** deux entiers naturels non nuls et **F** une fonction définie de la façon suivante :

F(a,0) = 1
F(a,n) = F(a*a , n div 2) Si **n** est **pair**
F(a,n) = a * F(a*a , (n-1) div 2) Si **n** est **impair**

1) Donner la trace d'exécution de la fonction **F** pour chacun des cas suivants :
   1^er cas : a = 2 et n = 2
   2^éme cas : a = 2 et n = 3
2) En déduire le rôle de la fonction **F**.
3) Ecrire un **algorithme récursif** de la fonction **F**.

Exercice 3 : (4 points)

En mathématiques, la constante de Brun (**B**) des nombres premiers jumeaux est la somme de la série des inverses des nombres premiers distants de 2.

$$B = \left(\frac{1}{3} + \frac{1}{5}\right) + \left(\frac{1}{5} + \frac{1}{7}\right) + \left(\frac{1}{11} + \frac{1}{13}\right) + \left(\frac{1}{17} + \frac{1}{19}\right) + \left(\frac{1}{29} + \frac{1}{31}\right) + \cdots$$

On rappelle qu'un nombre est dit premier s'il est divisible uniquement par **1** et par lui-même. Par convention l'entier **1** n'est pas premier.

Travail demandé :

Ecrire un algorithme d'une fonction **Brun** (*epsilon*) permettant de calculer, à *epsilon* prés, une valeur approchée de la constante de Brun définie précédemment (avec *epsilon* un réel passé en paramètres et dont la valeur est déjà saisie dans le module appelant).

NB : Chaque algorithme proposé doit être accompagné d'un tableau de déclaration des objets ayant la forme suivante :

Objet	Type / Nature	Rôle

Problème : (10 points)

La stéganographie est une méthode qui consiste à cacher un texte dans une image numérique bitmap. Cette méthode peut aider à échanger des messages secrets.

Dans une image **RVB**, chaque pixel est représenté par une chaîne de **6** chiffres hexadécimaux ; les deux premiers représentent l'intensité de la couleur "**Rouge**", les deux suivants celle de la couleur "**Vert**" et les deux derniers représentent l'intensité de la couleur "**Bleu**".

Exemples :

Couleur	Rouge	Vert	Noir	Blanc	Orange	Bleu	Rose
Code RVB	"FF0000"	"00FF00"	"000000"	"FFFFFF"	"ED7F10"	"0000FF"	"FD6C9E"

On se propose d'utiliser cette technique pour crypter un texte. Pour cela on dispose d'un fichier texte à crypter nommé "**source.txt**" situé sur la racine du disque C et comportant N lignes non vides ($\mathbf{1 \leq N \leq 40}$) de longueur maximale **120** caractères chacune.

Le procédé de cryptage est décrit ci-dessous :

- générer une matrice carrée **M (40x40)** à partir du fichier "**source.txt**" comme suit :
  - initialiser les cases de la matrice par le code de la couleur blanche "**FFFFFF**"
  - remplir chaque ligne de la matrice **M** par une ligne du fichier "**source.txt**" de la manière suivante :
    - ajouter à la fin de la ligne, si c'est nécessaire, un ou deux espaces pour que sa longueur soit divisible par trois
    - subdiviser la ligne en blocs de trois caractères consécutifs et remplir chaque case de la matrice par la chaîne résultante de la concaténation des équivalents hexadécimaux du code **ASCII** des trois caractères de chaque bloc
- générer un fichier "**code.txt**" à partir de la matrice **M**, où chaque ligne du fichier correspond à la concaténation du contenu d'une colonne de la matrice.

Exemple :

Pour le fichier "**source.txt**" suivant :

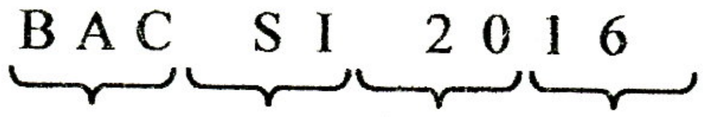
```
BAC SI 2016
44 %
BAC SC 2016
60 %
```

On obtient la matrice **M** suivante :

	1	2	3	4		39	40
1	424143	205349	203230	313620	FFFFFF	FFFFFF	FFFFFF
2	343420	252020	FFFFFF	FFFFFF	FFFFFF	FFFFFF	FFFFFF
3	424143	205343	203230	313620	FFFFFF	FFFFFF	FFFFFF
4	363020	252020	FFFFFF	FFFFFF	FFFFFF	FFFFFF	FFFFFF
	FFFFFF	FFFFFF	FFFFFF	FFFFFF	FFFFFF	FFFFFF	FFFFFF
	FFFFFF	FFFFFF	FFFFFF	FFFFFF	FFFFFF	FFFFFF	FFFFFF
39	FFFFFF	FFFFFF	FFFFFF	FFFFFF	FFFFFF	FFFFFF	FFFFFF
40	FFFFFF	FFFFFF	FFFFFF	FFFFFF	FFFFFF	FFFFFF	FFFFFF

En effet :

- la matrice **M** a été initialisée par le code de la couleur blanche "**FFFFFF**"
- la première ligne de la matrice **M** est remplie à partir de la première ligne du fichier "**source.txt**" comme suit :
  - étant donné que la longueur de la première ligne du fichier "**source.txt**" est non divisible par **3**, on ajoute un espace à la fin pour obtenir **4** groupements de **3** caractères consécutifs :



  - le premier élément de la matrice **M[1,1]** est égal à "**424143**" car :
    - le code **ASCII** de "**B**" est **66** et son équivalent hexadécimal est **42**
    - le code **ASCII** de "**A**" est **65** et son équivalent hexadécimal est **41**
    - le code **ASCII** de "**C**" est **67** et son équivalent hexadécimal est **43**
    - la concaténation des **3** équivalents hexadécimaux donne "**424143**" d'où le contenu de **M[1,1]** = "**424143**"
  - le même procédé donne **M[1,2]** = "**205349**", **M[1,3]** = "**203230**" et **M[1,4]** = "**313620**"
- les autres lignes de la matrice **M** sont remplies selon le même procédé.

D'où, en concaténant les valeurs de chaque colonne de la matrice **M** pour former une ligne du fichier, on obtient le fichier "**code.txt**" ci-dessous :

```
424143343420424143363020FFFFFF.....FFFFFFFFFFFF
205349252020205343252020FFFFFF.....FFFFFFFFFFFF
203230FFFFFF203230FFFFFFFFFFFF.....FFFFFFFFFFFF
313620FFFFFF313620FFFFFFFFFFFF.....FFFFFFFFFFFF
.....
FFFFFFFFFFFF.....FFFFFFFFFFFF
```

Travail demandé :

1- Analyser le problème en le décomposant en modules.
2- Ecrire un algorithme solution pour chaque module envisagé. Chaque algorithme proposé doit être accompagné d'un tableau de déclaration des objets ayant la forme suivante :

Objet	Type / Nature	Rôle

Corrigé : *Algorithmique et Programmation*
Section : Sciences de l'informatique
Session de contrôle 2017

Exercice 1 : (3 points)

Question n°1 :

P	Id=123	Id=113	Id=125	Id=115	Id=363	Id=263	Id=430	Id=380	Id=455	Id=663
	Age=57	Age=57	Age=22	Age=30	Age=35	Age=18	Age=33	Age=55	Age=23	Age=19
	Genre=M	Genre=F	Genre=M	Genre=F	Genre=M	Genre=F	Genre=M	Genre=F	Genre=F	Genre=F

Question n°2 :

Cette procédure permet de **fusionner** les deux tableaux H et F en **un tableau P** d'une manière **alternée** jusqu'à la fin du tableau ayant la plus petite taille ; **le reste des valeurs de l'autre tableau seront placées successivement à la fin de P**.

Question n°3 :

```
0) Def Proc Traitement (n1, n2 : Octet ; H, F : Tab1 ; Var k : Octet ; Var P : Tab2)
1) k←0, i ← 0, j←0
   Répéter
        k←k+1
        i ←i+1
        j ←j+1
        Si H[i].id < F[j].id Alors
                                P[k] ← H[i]
                                j ←j-1
                Sinon
                        P[k] ← F[j]
                        i←i-1
        Fin Si
   Jusqu'à (i=n1) ou (j=n2)
   Si (i=n1) Alors
            Pour c de j+1 à n2 Faire
                    k←k+1
                    P[k] ← F[c]
            FinPour
        Sinon
            Pour c de i+1 à n1 Faire
                    k←k+1
                    P[k] ← H[c]
            FinPour
    FinSi
2) Fin Traitement
```

Exercice 2 : (3 points)

Question n°1 :

F (2, 2) = F (4, 1) = 4 * F (16, 0) = 4 * 1 = 4
F (2, 3) = 2 * F (4, 1) = 2 * 4 * F (16, 0) = 2 * 4 * 1 = 8

Question n°2 :

Cette fonction retourne la **puissance** d'**ordre n** d'un entier a (a^n).

Question n° 3 :

```
0) Def FN F (a, n : Entier) : Entier Long
1) Si n = 0 Alors F← 1
        Sinon Si n mod 2 = 0 Alors F ← FN F (a*a, n div 2)
                Sinon F ← a * FN F (a*a, (n-1) div 2)
   FinSi
2) Fin F
```

Exercice 3 : (4 points)

L'algorithme de la fonction Brun :

```
0) Def FN Brun (Epsilon : Réel) : Réel
1) B← 0, k← 1
   Répéter
        k←k+2
        Si (FN Premier(k)) et (FN Premier (k+2)) Alors
                B1←B
                B←B+1/k + 1/(k+2)
        FinSi
   Jusqu'à abs (B-B1) < Epsilon
2) Brun ←B
3) Fin Brun
```

Tableau de déclaration des objets

Objet	Type / Nature	Rôle
K	Entier Long	Compteur d'entier impair
B, B1	Réel	Calculer la constante de Brun
Premier	Fonction	Vérifier si un entier est premier

L'algorithme de la fonction Premier :

```
0) Def  Fn Premier  (N : Entier Long) : Booléen
1) i←2
   TANTQUE (N mod i <>0) ET (i≤N div 2) FAIRE
        i←i+1
   FinTantque
2) Premier ← (i> N div 2) ET (N>1)
3) Fin Premier
```

Tableau de déclaration des objets

Objet	Type / Nature	Rôle
i	Entier Long	Compteur d'entier impair

Problème : (10 points)

Analyse du programme Principal :

```
Résultat = G
G = Associer (G,"Code.txt"),
     Proc Resultat(M,G)
M = Proc FormationMatrice(F,M)
F = Associer (F,"Source.txt")
```

Tableau de déclarations des nouveaux types

Type
Matrice = Tableau de 40 x 40 chaîne [6]

Tableau de déclarations des objets globaux

Objet	Type / Nature	Rôle
F	Texte	Fichier texte à crypter
G	Texte	Fichier crypté
M	Matrice	Matrice utilisée pour crypter F
Resultat	Procédure	Permet de générer le fichier crypté à partir de la matrice M
FormationMatrice	Procédure	Permet de remplir la matrice M à partir du fichier à crypter

Algorithme de la procédure FormationMatrice :

```
0) Def Proc FormationMatrice (Var F : Texte ; Var M : Matrice)
1) Pour L de 1 à 40 Faire
        Pour C de 1 à 40 Faire
            M[L,C] ← "FFFFFF"
        FinPour
   FinPour
2) Ouvrir(F), L ← 0
   Tant que Non (Fin-Fichier(F)) Faire
        Lire_nl(F,Lig)
        L←L+1
        Si Long(Lig) mod 3 = 1 Alors Lig←Lig+"␣␣"
            Sinon Si Long(Lig) mod 3 = 2 Alors Lig←Lig+"␣"
        FinSi
        C←0
        Répéter
            C←C+1
            M[L,C] ← Fn Hexa(Ord(Lig[1])) + Fn Hexa(Ord(Lig[2])) + Fn Hexa(Ord(Lig[3]))
            Efface (Lig, 1, 3)
        Jusqu'à Long (Lig)=0
   FinTantque
3) Fermer(F)
4) Fin FormationMatrice
```

Tableau de Déclarations des Objets Locaux

Objet	Type / Nature	Rôle
L	Octet	Compteur de lignes
C	Octet	Compteur de colonnes
Lig	Chaîne de caractères	Contient une ligne du fichier à crypter
Hexa	Fonction	Calculer l'équivalent hexadécimal d'un entier de deux chiffres

Algorithme de la fonction Hexa :

```
0) Def  Fn Hexa(k : Octet) : Chaîne
1) a ← k div 16
2) b ← k mod 16
3) Si a < 10 Alors Cha ← Chr (ORD ("0") + a)
        Sinon Cha ← Chr (ORD ("A") + a - 10)
   FinSi
4) Si b < 10 Alors Chb ← Chr (ORD ("0") + b)
        Sinon Chb ← Chr (ORD ("A") + b - 10)
   FinSi
5) Hexa ←Cha + Chb
6) Fin Hexa
```

Tableau de déclarations des objets locaux

Objet	Type / Nature	Rôle
a	Octet	Quotient de la division euclidienne d'un entier k Par 16
b	Octet	Reste de la division euclidienne d'un entier k Par 16
Cha	Caractère	Equivalent hexadécimal de a
Chb	Caractère	Equivalent hexadécimal de b

Algorithme de la procédure Résultat :

1) Def Proc Résultat (M : Matrice ; Var G : Texte)
2) Recréer(G)
   Pour C de 1 à 40 Faire
   Lig←""
   Pour L de 1 à 40 Faire
   Lig←Lig+M[L,C]
   FinPour
   Ecrire_nl(G,Lig)
   FinPour
   Fermer(G)
3) Fin Résultat

Tableau de déclarations des objets locaux

Objet	Type / Nature	Rôle
L	Octet	Compteur de lignes
C	Octet	Compteur de colonnes
Lig	Chaine de caractères	Contient la concaténation du contenu d'une colonne

RÉPUBLIQUE TUNISIENNE MINISTÈRE DE L'ÉDUCATION ***** EXAMEN DU BACCALAURÉAT	Épreuve : **ALGORITHMIQUE ET PROGRAMMATION**	
	Section : **Sciences de l'informatique**	
	Durée : **3H**	Coefficient : **2.25**
SESSION 2016	**Session principale**	

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.
Les réponses à l'exercice1 doivent être rédigées sur les pages 1/5 et 2/5 qui doivent être remises avec la copie.

Exercice 1: (3,25 points)

Soient les déclarations suivantes :

Tableau de déclaration des nouveaux types

Type
Date = Enregistrement jour : 1..31 mois : 1..12 annee : 2003..2010 **fin Date** **Eleve = Enregistrement** nom : chaîne[15] moyenne : réel dat_nais : **Date** **fin Eleve** **Feleve = fichier d'Eleve**

Tableau de déclaration des objets

Objet	Type/Nature
F_El	Feleve
E	Eleve

Travail à faire :

1- On se propose d'ajouter dans le fichier **F_El** la liste des élèves présentée dans le tableau ci-dessous.

Compléter la deuxième colonne du tableau par :

- **V** dans le cas où les données relatives à l'élève ne présentent aucune anomalie.
- **F** dans le cas contraire, tout en justifiant la réponse dans la troisième colonne.

Elève	V/F	Justification
"Mohamed", 17, "12/11/2000"		
"Kefi", 14.5, "15/02/2003"		
"Ali", 16, "16/13/2009"		

2- Remplir le tableau suivant par les séquences d'instructions algorithmiques, permettant de réaliser les traitements indiqués dans la première colonne, sachant que le fichier **F_El** est initialement ouvert et son pointeur est positionné sur le premier enregistrement :

Traitement	**Séquences d'instructions**
Afficher le premier enregistrement du fichier **F_El**.	
Ajouter l'élève ("Tounsi", 14.5, "15/02/2004") à la fin du fichier **F_El**.	
Ajouter un point (1) à la moyenne du deuxième élève du fichier **F_El**.	

RÉPUBLIQUE TUNISIENNE MINISTÈRE DE L'ÉDUCATION ***** EXAMEN DU BACCALAURÉAT	Épreuve : **ALGORITHMIQUE ET PROGRAMMATION**	
	Section : **Sciences de l'informatique**	
	Durée : **3H**	Coefficient : **2.25**
SESSION 2016	**Session principale**	

Exercice 2 : (3,75 points)

Soit **M** une matrice carrée d'ordre **N** (avec $4 \leq N \leq 10$), représentant une grille de jeu dans laquelle deux joueurs marquent des cases, à tour de rôle. Chaque case de cette matrice peut contenir soit 0, soit 1, soit 2.

- 0 : indique que cette case n'a pas été marquée.
- 1 : indique que cette case est marquée par le premier joueur.
- 2 : indique que cette case est marquée par le deuxième joueur.

Le jeu s'arrête :

- lorsqu'un joueur marque quatre (4) cases consécutives dans le sens horizontal ou dans le sens vertical. Ce joueur sera déclaré gagnant.
- lorsque toutes les cases sont marquées sans qu'aucun joueur n'arrive à assembler quatre (4) cases consécutives dans le sens horizontal ou dans le sens vertical. Dans ce cas, la partie est considérée nulle.

Exemple : Pour la matrice **M** d'ordre 6 suivante :

0	0	1	0	0	0
0	0	2	0	0	0
0	1	2	1	0	0
0	2	2	1	0	0
1	2	2	2	1	0
1	2	1	1	2	0

Le 2ème joueur est gagnant car il a marqué quatre cases consécutives dans le sens vertical.

Travail à faire :

Ecrire un algorithme d'un module intitulé "**Gagnant**" qui, à partir d'une matrice **M** d'ordre **N** et représentant une grille de jeu marquée par deux joueurs, affiche le résultat du jeu (premier joueur gagnant ou deuxième joueur gagnant ou partie nulle).

NB : M et **N** sont déjà saisis au niveau du programme appelant.

Exercice 3 : (4,5 points)

Un RIB est un code d'identification bancaire délivré par une banque à un titulaire de compte bancaire. Il est composé de 4 champs :

- **CB** (code de la banque) : 2 chiffres
- **CA** (code de l'agence) : 3 chiffres
- **NC** (numéro du compte) : 13 chiffres
- **CR** (clé RIB) : 2 chiffres

XX	XXX	XXXXXXXXXXXXX	XX
CB	**CA**	**NC**	**CR**

Pour calculer la clé RIB, on procède comme suit :

- Multiplier **N** par 100, sachant que **N** est un nombre de 18 chiffres obtenu suite à la concaténation des chiffres des champs **CB**, **CA** et **NC**.
- Calculer le reste de la division entière du nombre ainsi obtenu par 97.
- Soustraire de 97, le reste obtenu dans l'étape précédente. Le résultat de cette soustraction représente la clé de contrôle dite clé RIB, qui ne peut prendre qu'une valeur entre 01 et 97.

Un RIB est valide lorsque la clé figurant dans le RIB est égale à celle calculée en utilisant la méthode de calcul décrite ci-dessus.

Exemple :

Pour le RIB suivant :

10	407	02400675324 81	20

- N est égal à 10407024006753248 1
  N * 100 est égal à 1040702400675324810 0
- Le reste de la division entière de N par 97 est égal à 77
- 97 – 77 = 20 qui représente la clé de contrôle dite clé RIB.
  D'où, le RIB est valide puisque la clé de contrôle calculée est égale à la clé figurant dans le RIB.

Travail à faire :

Ecrire une analyse d'un module intitulé "**TRIB**" qui, à partir d'un fichier "**RIB.txt**" contenant des codes RIB, à raison d'un code par ligne, permet de :

- Remplir un fichier "**RIB_valide.txt**" par les RIB valides du fichier "**RIB.txt**".
- Trier le fichier "**RIB_valide.txt**" selon l'ordre croissant du code de la banque.

NB :

- Le candidat n'est pas appelé à remplir le fichier "**RIB.txt**".
- Le fichier "**RIB_valide.txt**" sera enregistré sur la racine du disque C.
- Le candidat peut utiliser une fonction intitulée **Mod97(CH)** ayant comme paramètre une chaîne de caractères **CH** représentant une valeur numérique très grande **N** et qui permet de retourner le reste de la division entière de **N** par **97**. Le candidat n'est pas appelé à développer cette fonction.

Exercice 4 : (8,5 points)

Le **codage de Fibonacci** est un code binaire, utilisant des termes de la suite de Fibonacci et servant essentiellement dans la compression de données.

Pour déterminer le code de **Fibonacci** d'un entier **K** strictement positif, on suit les étapes suivantes :

Etape 1 : Déterminer la liste des termes de la suite de Fibonacci inférieurs ou égaux à **K**, sachant que la suite de Fibonacci **U** est définie comme suit :

$$\begin{cases} U_1 = 1, U_2 = 1 \\ U_n = U_{n-1} + U_{n-2} \quad \text{pour } n > 2 \end{cases}$$

Etape 2 : Décomposer l'entier **K** en une somme des termes de la suite de Fibonacci déjà calculés dans l'étape précédente, tout en commençant par utiliser le plus grand terme inférieur à **K** et sans prendre en considération le premier terme de la suite (U_1).

Etape 3 : Former un code binaire à partir de la liste des termes calculée dans l'étape 1 et sans prendre en considération le premier terme de la suite (U_1) : en concaténant le caractère "1" dans le cas où le terme a été utilisé dans la somme calculée au niveau de l'étape 2 et en concaténant le caractère "0" dans le cas contraire.

Etape 4 : Ajouter à la fin du code obtenu précédemment le caractère "1" pour obtenir le code de Fibonacci.

On se propose d'écrire un programme qui permet de saisir un entier **K** strictement positif et d'afficher le code de Fibonacci correspondant.

Exemple : Pour **K** = 50,

Etape 1 : La liste des termes de la suite de Fibonacci inférieurs ou égaux à 50 et sans prendre en considération le premier terme de la suite (U_1) est : 1, 2, 3, 5, 8, 13, 21 et 34.

Etape 2 : La décomposition de **K** sera comme suit : 50 = 34 + 13 + 3

Etape 3 : Etant donnée la liste des termes obtenue dans l'étape 1 et sans prendre en considération le premier terme de la suite (U_1) : 1, 2, 3, 5, 8, 13, 21 et 34
En concaténant le caractère "0" pour les termes 1, 2, 5, 8 et 21 qui n'ont pas été utilisés dans la somme et en concaténant le caractère "1" pour les termes 3, 13 et 34 qui ont été utilisés dans la somme, le code binaire formé est : 00100101

Etape 4 : En ajoutant à la fin du code binaire obtenu précédemment le caractère "1", le code de Fibonacci est : **001001011**

Travail à faire :

1- Analyser le problème en le décomposant en modules.

2- Analyser chacun des modules envisagés.

RÉPUBLIQUE TUNISIENNE MINISTÈRE DE L'ÉDUCATION ※※※※ **EXAMEN DU BACCALAURÉAT** **SESSION 2016**	Épreuve : ALGORITHMIQUE ET PROGRAMMATION **Corrigé**
Section : Sciences de l'informatique	Session principale

Exercice 1: (3 + 3.5 = 6.5 points)

1-

Elève	V/F	Justification	*Nombre de points*
"Mohamed",17,"12/11/2000"	*F*	*Valeur incompatible avec le type Date.*	*1*
"Kefi", 14.5, "15/02/2003"	*F*	*Valeur incompatible avec le type Date.*	*1*
"Ali", 16, "16/13/2009"	*F*	*Valeur incompatible avec le type Date.*	*1*

Elève	V/F	Justification	*Nombre de points*
"Mohamed",17,"12/11/2000"	*F*	*L'année doit être comprise entre 2003 et 2010.*	*1*
"Kefi", 14.5, "15/02/2003"	*V*		*1*
"Ali", 16, "16/13/2009"	*F*	*Le mois doit être compris entre 1 et 12*	*1*

2-

Traitement	Séquences d'instructions	*Nombre de points*
Afficher le premier enregistrement du fichier **F_El**.	*Lire (F_El, el)* *Ecrire(el.nom,el.moyenne,el.date_nais.jour,"/",el.date_nais.mois,"/",el.date_nais.annee)*	*0.25* *0.25*
Ajouter l'élève ("Tounsi", 14.5,"15/02/2004") à la fin du fichier **F_El**.	*Pointer (F_El,Taille_fichier(F_El))* *e.nom ←"Tounsi"* *e.moyenne ← 14.5* *e.date_nais.jour ← 15* *e.date_nais.mois ← 2* *e.date_nais.annee ← 2004* *Ecrire (F_El, e)*	*0.25* *0.25* *0.25* *0.25* *0.25* *0.25* *0.25*
Ajouter un point à la moyenne du deuxième élève du fichier **F_El**.	*Pointer (F_El, 1)* *Lire (F_El,e)* *e.moyenne ← e.moyenne + 1* *Pointer (F_El, 1)* *Ecrire (F_El, e)*	*0.25* *0.25* *0.25* *0.25* *0.25*

Exercice 2 : (7.5 points)

```
0) DEF PROC Gagnant (M:Tab ; N:Entier)
1) Si FN trouve(M,N)=2 Alors Ecrire("Joueur 2 gagnant")
   Sinon  Si FN  trouve(M,N)=1 Alors Ecrire("Joueur 1 gagnant")
          Sinon Ecrire("Partie nulle")
    FinSi
2) Fin PROC Gagnant

0) DEF FN trouve (M:Tab ; N:Entier): Entier
1) T ← 0
   SI FN Horiz_verti(M,N,1) ≠ 0 Alors T ←FN  Horiz_verti(M,N,1)
   FinSi
   SI FN Horiz_verti(M,N,2) ≠ 0 Alors T ← FN Horiz_verti(M,N,2)
   FinSi
2)  Trouve ← T
3) Fin FN Trouve

0) DEF FN Horiz_verti (M:Tab; N,joueur: Entier): Entier
1) J_G ← 0
   i←0
   Répéter
      i←i+1
      j←0
     Répéter
       j←j+1
       Si (M[i,j]=joueur) ET (M[i,j+1]=joueur) ET (M[i,j+2]=joueur) ET (M[i,j+3]=joueur)
             Alors   J_G ← joueur
       FinSi
       SI (M[j,i]=joueur) ET (M[j+1,i]=joueur) ET (M[j+2,i]=joueur) ET (M[j+3,i]=joueur)
             Alors   J_G ← joueur
       FinSi
    Jusqu'à (J_G=joueur) ou (j=N-3)
   Jusqu'à (J_G=joueur) ou (i=N)
2)  Horiz_verti ← J_G
3) Fin FN Horiz_verti
```

Barème :

Traitement	*Nombre de points*
Test + affichage du résultat	*(0.25+0.25)*3*
Vérification horizontale (parcours + test)	*1+1*
Vérification verticale (parcours + test)	*1+1*
Retour du résultat de la vérification	*0.75+0.75*
Entêtes des modules	*0.5*

Exercice 3 : (9 points)

```
DEF PROC RIB (Var f , f_v : Texte)
Résultat = f_v
f_v =  PROC Remplir_RIB_valide(f,f_v)
       PROC Tri(f_v)
Fin PROC RIB
```

```
DEF PROC Remplir_RIB_valide (Var f,f_v : texte)
Résultat = f_v
f_v = [Associer (f_v,"c:\RIB_valide.txt"),
       Recréer (f_v), Ouvrir (f), Ouvrir (f_v) ]
       Tantque  Non Fin_fichier (f) Faire
          Lire_nl (f,R)
          cle ← Sous_chaîne(R,19,2)
          valeur (cle,cl,e);
          code ← Sous_chaîne(R,1,18)+"00"
          cr  ← 97 – FN Mod97(code)
          SI cr = cl Alors Ecrire_nl (f_v,R)
          FinSI
       FinTantque
       Fermer (f)
       Fermer (f_v)
Fin PROC Remplir_RIB_ valide
```

Tableau de déclaration des objets locaux

Objet	*Type / Nature*
R, cle, code	*Chaîne de caractères*
Cl, e, n, cr	*Entier*
Mod97	*Fonction*

```
DEF PROC Tri (var f_v:texte)
Résultat = f_v
F_v = [Ouvrir (f_v), N ←0 ]
       {transfert dans un tableau}
       Tantque Non Fin_ficher (f_v) Faire
          N ← N+1
          Lire_nl (f_v, T[N])
       Fin Tantque
       Pour i de 1 à N-1 Faire
          ind ← 1
          cbi← Sous_chaîne(t[i],1,2)
          Pour j de 2 à N Faire   {tri du tableau}
            cbj ← Sous_chaîne (t[j],1,2)
            SI cbi > cbj Alors ind←j
            FinSi
          FinPour
          temp←t[i]
          t[i]←t[ind]
          t[ind]←temp
       FinPour
       Ouvrir (f_v)     {transfert dans le fichier}
       Pour i de 1 à N Faire
          Ecrire_nl (f_v,t[i])
      Finpour
     Fermer (f_v)
Fin PROC Tri
```

Tableau de déclaration des objets locaux

Objet	*Type / Nature*
cbi, cbj, temp	*Chaîne de caractères*
N, i, ind, j	*Entier*
T	*Tableau de 100 chaîne [20]*

Barème :

Traitement	Nombre de points
Remplissage du fichier "RIB_valide.txt": - *Association + Création + Ouverture + Fermeture des fichiers* - *Parcours du fichier "RIB.txt" + Lecture d'une ligne* - *Calcul de la clé* - *Test + Ecriture*	 *0.25*5* *1 + 0.5* *1* *0.5+0.5*
Tri du fichier "RIB_Valide.txt" : - *Transfert dans un tableau* - *Tri du tableau* - *Transfert des éléments du tableau trié dans le fichier*	 *1* *1* *1*
Entêtes des modules	*0.5*
TDOL	*0.75*

Exercice 4 : (17 points)

1) Analyse du programme principal :

Nom : Codage_Fibo
Résultat = PROC Affiche_code(M,n)
(M,n) = PROC Terme_fib(k,n,M)
PROC Coeff_terme(k,n,M)
K = PROC Saisie(k)
Fin Codage_Fibo

Tableau de déclaration des nouveaux types

Type
Tab = Matrice de 2 lignes et de 100 colonnes d'entiers

Tableau de déclaration des objets globaux

Objet	Type / Nature
M	*Tab*
K,n	*Entier*
Affiche_code	*Procédure*
Terme_Fibo	*Procédure*
Coeff_terme	*Procédure*
Saisie	*Procédure*

2) Analyse des modules envisagés

DEF PROC affiche_code(M:Tab;n:entier)
Résultat = Ecrire(ch)
ch = [ch←""]
Pour i de 2 à n Faire
Convch(M[2,i],c)
ch ← ch + c
Fin Pour
ch←ch + "1"
Fin PROC affiche_code

Tableau de déclaration des objets locaux

Objet	Type / Nature
Ch	*Chaîne de caractères*
i	*Entier*
c	*Caractère*

DEF PROC terme_fib(k:entier ; var n:entier;var M:tab)
Résultat = M,n
(M,n) = [M[1,1]←1, M[1,2]←1, n ←2]
Tantque M[1,n]+ M[1,n-1]<= k Faire
n←n+1
M[1,n] ← M[1,n-1]+ M[1,n-2]
Fin Tantque
Fin PROC terme_fib

DEF PROC coeff_terme(k,n:entier ; var M:tab)

Résultat = M
M= [s←0]
Si M[1,n]=k alors n←n-1
Finsi
Pour i de n à 2 pas -1 Faire
SI (s + M[1,i]) ≤ k Alors
s← s + M[1,i]
M[2,i]←1
Sinon M[2,i]←0
FinSI
FinPour
Fin PROC coeff_terme

Tableau de déclaration des objets locaux

Objet	***Type / Nature***
i, s	*Entier*

DEF PROC Saisie(var k:entier)

Résultat = k
k= [ ]
Répéter
k=donnée
Jusqu'à k > 0
Fin PROC Saisie

Barème :

Traitement	***Nombre de points***
Décomposition	*1*
Appels + cohérence des paramètres	*0.5 + 0.5*
Saisie de k avec respect des contraintes	*1.5 = 0.5 + 1*
Détermination des termes de Fibonacci - *Initialisation* - *Parcours* - *Calcul du nouveau terme* - *Incrémentation du compteur*	 *0.75* *1* *1* *0.25*
Décomposition de K en une somme des termes de la suite de Fibonacci - *Initialisation* - *Parcours* - *Test* - *Calcul de somme*	 *0.5* *1.5* *1* *1*
Détermination du code de Fibonacci + Affichage - *Initialisation* - *Parcours* - *Test (0 ou 1)* - *Concaténation* - *Ajout du caractère "1"* - *Affichage*	 *0.5* *1* *0.5* *0.5* *0.5* *1*
TDNT + TDOG + TDOL	*0.5 + 1 + 1*

RÉPUBLIQUE TUNISIENNE MINISTÈRE DE L'ÉDUCATION ***** EXAMEN DU BACCALAURÉAT	Épreuve : **ALGORITHMIQUE ET PROGRAMMATION**	
	Section : **Sciences de l'informatique**	
	Durée : **3H**	Coefficient : **2.25**
SESSION 2016	**Session de contrôle**	

Exercice 1 : (3 points)

Soit une fonction **Symetrie** qui vérifie si le contenu d'un fichier d'entiers **F**, déjà rempli, est symétrique.
On propose ci-dessous un algorithme de cette fonction, contenant trois erreurs dans le choix des fonctions et des procédures prédéfinies utilisées :

```
0) DEF FN Symetrie (Var F : fiche_ent) : booléen
1) Recréer(F) ; S ← Vrai ; t ← taille_fichier(F)
   Pour i de 1 à fin_fichier(F) div 2 Faire
      Pointer(F,i)
      Lire_nl(F,M)
      Pointer(F,t – i + 1)
      Lire(F,N)
      S ← (S et (M =N))
   Fin Pour
   Fermer(F)
2) Symetrie ← S
3) Fin Symetrie
```

Travail à faire :

1) Compléter le tableau ci-dessous en remplissant la première colonne par les fonctions ou les procédures prédéfinies dont l'utilisation est erronée et la deuxième colonne par les fonctions ou les procédures prédéfinies adéquates :

Fonction ou procédure prédéfinie dont l'utilisation est erronée	**Fonction ou procédure prédéfinie adéquate**

2) Dans le but d'améliorer l'algorithme proposé, réécrire la séquence n°1 en utilisant la structure itérative adéquate et en tenant compte des corrections apportées dans la question n°1.

Exercice 2 : (4,75 points)

Soit la suite de Perrin définie par :

$$\begin{cases} U_0 = 3,\ U_1 = 0\ ,\ U_2 = 2 \\ U_n = U_{n-2} + U_{n-3} \ \text{ pour } n \geq 3 \end{cases}$$

Cette suite vérifie la proprieté suivante : **"Pour tout entier n > 2, si n divise U_n alors n est un nombre premier"**.

Travail à faire :

1- Ecrire une analyse d'un module intitulé **"Verif_pr"**, permettant de vérifier si un entier **N** est premier et ce, en utilisant la propriété décrite précédemment.

NB : **N** est déjà saisi dans le programme appelant.

2- Ecrire un algorithme d'un module récursif intitulé **"Verif_geo"**, permettant de vérifier si la suite **U** est géométrique sur un intervalle [a, b] donné.

NB :

- Une suite **U** est dite géométrique s'il existe un réel **q** (appelé raison) tel que pour tout entier **n** de l'intervalle [a, b], $\mathbf{U_{n+1} = q \times U_n}$.
- **a** et **b** sont déjà saisis au niveau du programme appelant (avec $b \geq a + 2$).

Exercice 3 : (3,5 points)

Un **nombre primaire**, également appelé puissance première, est une puissance à exposant entier positif non nul d'un nombre premier.

Exemples :

- 5, 9 et 16 sont des nombres primaires, car $5 = 5^1$, $9 = 3^2$ et $16 = 2^4$.
- 6 et 36 ne sont pas des nombres primaires, car on ne peut pas les écrire sous forme d'une puissance à exposant entier positif non nul d'un nombre premier.

Travail à faire :

Ecrire un algorithme d'un module intitulé **"Nb_primaire"** qui affiche les **N** premiers nombres primaires.

NB : **N** est déjà saisi au niveau du programme appelant.

Exercice 4 : (3,5 points)

Soit un tableau **T_DN** contenant les dates de naissance de **N** personnes. On se propose de trier le tableau **T_DN** par ordre croissant.

Travail à faire :

1- Donner une structure de données adéquate pour représenter une date de naissance.
2- En utilisant la structure de donnée proposée dans la question n°1, écrire un algorithme d'un module intitulé "**Tri**" qui permet de trier les **N** éléments du tableau **T_DN** par ordre croissant de la date de naissance.

NB: **N** est déjà saisi au niveau du programme appelant.

Exercice 5 : (5,25 points)

Soit **M** une matrice carrée d'ordre **N** (avec $N \leq 15$) remplie par des lettres majuscules. On se propose de créer, sur la racine du disque C, un fichier "**Symetrie.txt**" formé par les lignes et les colonnes symétriques se trouvant dans la matrice **M** ainsi que leurs nombres. Pour cela, on procède comme suit :

- La première ligne du fichier contient les contenus des lignes symétriques de la matrice **M**, séparés par le caractère "*".
- La deuxième ligne du fichier contient le nombre de lignes symétriques contenues dans la matrice **M**.
- La troisième ligne du fichier contient les contenus des colonnes symétriques de la matrice **M**, séparés par le caractère "*".
- La quatrième ligne du fichier contient le nombre de colonnes symétriques contenues dans la matrice **M**.

NB : Une ligne ou une colonne d'une matrice est dite symétrique si la concaténation des caractères contenus dans ses cases forme une chaîne palindrome.

Exemple :

Pour **N** = 5 et la matrice **M** suivante :

G	B	E	B	G
N	A	R	O	U
E	I	M	L	C
M	A	L	A	M
O	B	E	W	G

- La ligne "MALAM" comme indiquée ci-contre est un exemple de lignes symétriques de la matrice **M**.
- La colonne "BAIAB" comme indiquée ci-contre est un exemple de colonnes symétriques de la matrice **M**.

Le fichier "**Symetrie.txt**" aura le contenu suivant :

```
GBEBG*MALAM
2
BAIAB
1
```

Travail à faire :

Ecrire un algorithme d'un module intitulé "**L_C_Sym**" permettant de remplir le fichier "**Symetrie.txt**" comme décrit précédemment.

NB: **M** et **N** sont déjà saisis au niveau du programme appelant.

RÉPUBLIQUE TUNISIENNE MINISTÈRE DE L'ÉDUCATION ***** EXAMEN DU BACCALAURÉAT	Épreuve : **ALGORITHMIQUE ET PROGRAMMATION** Section : **Sciences de l'informatique** Durée : **3H** — Coefficient : **2.25**
SESSION 2016	**Session de contrôle**

Section : ……..…….. N° d'inscription : …… Série : ……...
Nom et prénom : ……..…..
Date et lieu de naissance : ……..……..……..…..

Signatures des surveillants
……
……

Le sujet comporte 3 pages numérotées de 1/3 à 3/3.
Les réponses à la question 1 de l'exercice 1 doivent être rédigées sur la page 1/3 qui doit être remise avec la copie.

Exercice 1 : (3 points)

Soit une fonction **Symetrie** qui vérifie si le contenu d'un fichier d'entiers **F**, déjà rempli, est symétrique.
On propose ci-dessous un algorithme de cette fonction, contenant trois erreurs dans le choix des fonctions et des procédures prédéfinies utilisées :

```
0) DEF FN Symetrie (Var F : fiche_ent) : booléen
1) Recréer(F) ; S ← Vrai ; t ← taille_fichier(F)
   Pour i de 1 à fin_fichier(F) div 2 Faire
      Pointer(F,i)
      Lire_nl(F,M)
      Pointer(F,t – i + 1)
      Lire(F,N)
      S ← (S et (M =N))
   Fin Pour
   Fermer(F)
2) Symetrie ← S
3) Fin Symetrie
```

Travail à faire :

1) Compléter le tableau ci-dessous en remplissant la première colonne par les fonctions ou les procédures prédéfinies dont l'utilisation est erronée et la deuxième colonne par les fonctions ou les procédures prédéfinies adéquates :

Fonction ou procédure prédéfinie dont l'utilisation est erronée	**Fonction ou procédure prédéfinie adéquate**

2) Dans le but d'améliorer l'algorithme proposé, réécrire la séquence n°1 en utilisant la structure itérative adéquate et en tenant compte des corrections apportées dans la question n°1.

Exercice 2 : (4,75 points)

Soit la suite de Perrin définie par :

$$\begin{cases} U_0 = 3,\ U_1 = 0\ ,\ U_2 = 2 \\ U_n = U_{n-2} + U_{n-3} \ \text{ pour } n \geq 3 \end{cases}$$

Cette suite vérifie la proprieté suivante : "**Pour tout entier n > 2, si n divise U_n alors n est un nombre premier**".

Travail à faire :

1- Ecrire une analyse d'un module intitulé "**Verif_pr**", permettant de vérifier si un entier **N** est premier et ce, en utilisant la propriété décrite précédemment.

NB : **N** est déjà saisi dans le programme appelant.

2- Ecrire un algorithme d'un module récursif intitulé "**Verif_geo**", permettant de vérifier si la suite **U** est géométrique sur un intervalle [a, b] donné.

NB :

- Une suite **U** est dite géométrique s'il existe un réel **q** (appelé raison) tel que pour tout entier **n** de l'intervalle [a, b], $\mathbf{U_{n+1} = q \times U_n}$.
- **a** et **b** sont déjà saisis au niveau du programme appelant (avec $b \geq a + 2$).

Exercice 3 : (3,5 points)

Un **nombre primaire**, également appelé puissance première, est une puissance à exposant entier positif non nul d'un nombre premier.

Exemples :

- 5, 9 et 16 sont des nombres primaires, car $5 = 5^1$, $9 = 3^2$ et $16 = 2^4$.
- 6 et 36 ne sont pas des nombres primaires, car on ne peut pas les écrire sous forme d'une puissance à exposant entier positif non nul d'un nombre premier.

Travail à faire :

Ecrire un algorithme d'un module intitulé "**Nb_primaire**" qui affiche les **N** premiers nombres primaires.

NB : **N** est déjà saisi au niveau du programme appelant.

Exercice 4 : (3,5 points)

Soit un tableau **T_DN** contenant les dates de naissance de **N** personnes. On se propose de trier le tableau **T_DN** par ordre croissant.

Travail à faire :

1- Donner une structure de données adéquate pour représenter une date de naissance.
2- En utilisant la structure de donnée proposée dans la question n°1, écrire un algorithme d'un module intitulé "**Tri**" qui permet de trier les **N** éléments du tableau **T_DN** par ordre croissant de la date de naissance.

NB: **N** est déjà saisi au niveau du programme appelant.

Exercice 5 : (5,25 points)

Soit **M** une matrice carrée d'ordre **N** (avec N ≤ 15) remplie par des lettres majuscules. On se propose de créer, sur la racine du disque C, un fichier "**Symetrie.txt**" formé par les lignes et les colonnes symétriques se trouvant dans la matrice **M** ainsi que leurs nombres. Pour cela, on procède comme suit :

- La première ligne du fichier contient les contenus des lignes symétriques de la matrice **M**, séparés par le caractère "*".
- La deuxième ligne du fichier contient le nombre de lignes symétriques contenues dans la matrice **M**.
- La troisième ligne du fichier contient les contenus des colonnes symétriques de la matrice **M**, séparés par le caractère "*".
- La quatrième ligne du fichier contient le nombre de colonnes symétriques contenues dans la matrice **M**.

NB : Une ligne ou une colonne d'une matrice est dite symétrique si la concaténation des caractères contenus dans ses cases forme une chaîne palindrome.

Exemple :

Pour **N** = 5 et la matrice **M** suivante :

G	**B**	**E**	**B**	**G**
N	**A**	**R**	**O**	**U**
E	**I**	**M**	**L**	**C**
M	**A**	**L**	**A**	**M**
O	**B**	**E**	**W**	**G**

- La ligne "MALAM" comme indiquée ci-contre est un exemple de lignes symétriques de la matrice **M**.
- La colonne "BAIAB" comme indiquée ci-contre est un exemple de colonnes symétriques de la matrice **M**.

Le fichier "**Symetrie.txt**" aura le contenu suivant :

GBEBG*MALAM
2
BAIAB
1

Travail à faire :

Ecrire un algorithme d'un module intitulé "**L_C_Sym**" permettant de remplir le fichier "**Symetrie.txt**" comme décrit précédemment.

NB: **M** et **N** sont déjà saisis au niveau du programme appelant.

RÉPUBLIQUE TUNISIENNE MINISTÈRE DE L'ÉDUCATION ◈◈◈◈ **EXAMEN DU BACCALAURÉAT** **SESSION 2015**	Épreuve : **ALGORITHMIQUE ET PROGRAMMATION**
	Durée : **3 h**
	Coefficient : **2,25**
Section : **Sciences de l'informatique**	**Session principale**

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

Nom et prénom : .....

Date et lieu de naissance : .....

Signatures des surveillants

.....

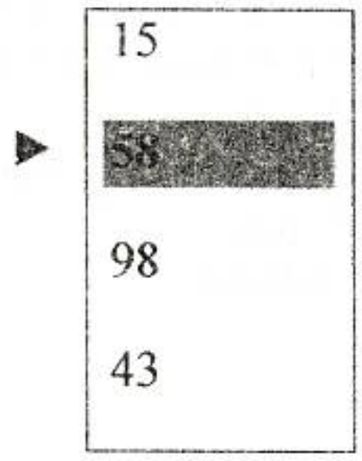
.....

Le sujet comporte 4 pages numérotées de 1/4 à 4/4.
Les réponses à l'exercice1 doivent être rédigées sur les pages 1/4 et 2/4
qui doivent être remises avec la copie

Exercice 1 (5,25 points)

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

1) Soit un fichier d'entiers, ayant comme nom logique **F.** On suppose que le pointeur est positionné sur le deuxième entier comme indiqué ci-dessous.



N.B : Toutes les instructions données dans a), b) et c) sont valides.

a) Le contenu de **X** après l'exécution de l'instruction **Lire(F , X)** est :

☐ 15 ☐ 58 ☐ 98

b) L'instruction **Pointer(F , 3)** permet de positionner le pointeur sur l'entier :

☐ 58 ☐ 98 ☐ 43

c) Le contenu de **Y** après l'exécution de l'instruction **Y ← Fin_fichier(F)** est :

☐ 43 ☐ 4 ☐ faux

Ne rien écrire ici

2) Soit la suite **U** définie par: $\begin{cases} U_0 = 1 \\ U_n = 2*U_{n-1} + n \end{cases}$ (avec **n** un entier supérieur ou égal à 1)

a) **U** est une suite récurrente d'ordre :

☐ 1 ☐ 2 ☐ 5

b) Le 3ème terme de la suite **U** (U_2) est égal à :

☐ 5 ☐ 8 ☐ 9

c) L'algorithme permettant de calculer U_n (avec $n \geq 1$) est :

☐
```
0)Def FN terme (n : entier) : entier
1) t[0] ← 1
2) Pour i de 1 à n faire
     t[i] ← 2*t[i-1]+n
   Fin pour
3) terme ← t[n]
4) Fin terme
```

☐
```
0)Def FN terme(n : entier) : entier
1) Si n=0 alors terme ← 1
     Sinon
       terme←2*FN terme(n-1)+n
     Fin si
2) Fin terme
```

☐
```
0)Def FN terme(n : entier) : entier
1) Up ← 1
2) Pour i de 2 à n faire
       Up ← 2*Up+i
   Fin pour
3) terme ← Up
4) Fin terme
```

Exercice 2 (3 points)

En arithmétique, un **auto-nombre** est un entier naturel **N** qui ne peut pas s'écrire sous la forme d'un nombre **M** ajouté à la somme des chiffres de **M**.

Exemples :

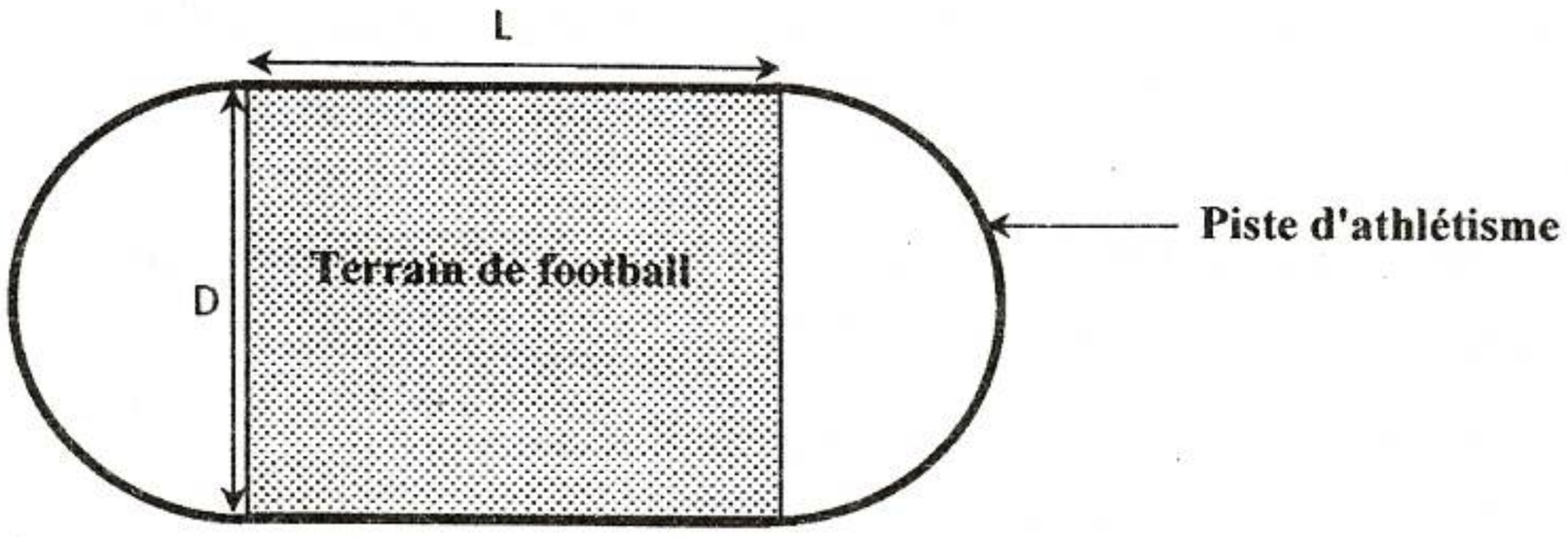
- Pour **N = 21**,
  N n'est pas un auto-nombre, puisqu'il peut être généré à partir de la somme d'un nombre **M** égal à **15** et les chiffres qui le constituent (**1** et **5**) c'est-à-dire **21 = 15 + 1 + 5**.
- Pour **N = 20**,
  N est un auto-nombre puisqu'il ne peut pas être généré à partir de la somme d'un nombre **M** et les chiffres qui le constituent.

Travail demandé :

Ecrire une **analyse** d'un module intitulé **Verif_auto_nombre**, permettant de **vérifier** si un entier naturel **N** strictement positif est un **auto-nombre**, sachant que N est déjà saisi dans l'analyse du programme principal.

Exercice 3 (3 points)

La direction d'une association sportive veut construire un stade formé par une piste d'athlétisme et un terrain de football, tout en cherchant à **maximiser la surface** de ce dernier.
Le terrain de football est un rectangle de longueur **L**, de largeur **D** et de surface **S**.
La piste d'athlétisme est de longueur **P** et formée par les deux arrêtes parallèles du terrain de football (de longueur **2 * L**) et les deux demi-cercles de diamètre **D** (de longueur π * **D**), comme le montre le schéma suivant :



Puisque **S = L * D** et **P = 2 * L + π * D** alors **S = L * (P – 2 * L) / π**

Travail demandé :

Etant donné que **L** varie de **0** à **P/2**, écrire un **algorithme** d'une fonction qui permet de déterminer, à **ε** près, la longueur optimale **Lopt** correspondante à la surface maximale **Smax** du terrain, sachant que **ε** et **P** sont déjà saisis dans l'algorithme du programme principal.

Problème (8,75 points)

Un administrateur d'un site web veut assurer un maximum de sécurité pour les utilisateurs du site. Pour ceci il décide de réaliser une application qui évalue la force des mots de passe des différents utilisateurs du site, sachant qu'un mot de passe est une chaîne de caractères qui ne comporte pas d'espaces et de lettres accentuées.

La force d'un mot de passe varie, selon la valeur d'un score calculé, de **"Très faible"** jusqu'à **"Très fort"** :

- Si le score < 20, la force du mot de passe est **"Très faible"**
- Sinon si le score < 40, la force du mot de passe est **"Faible"**
- Sinon si le score < 60, la force du mot de passe est **"Moyen"**
- Sinon si le score < 80, la force du mot de passe est **"Fort"**
- Sinon la force du mot de passe est **"Très fort"**

Le score se calcule en additionnant des bonus et en retranchant des pénalités.

Les bonus attribués sont :

- Nombre total de caractères * 4
- (Nombre total de caractères – nombre de lettres majuscules) * 2
- (Nombre total de caractères – nombre de lettres minuscules) * 3
- Nombre de caractères non alphabétiques * 5

Les pénalités imposées sont :

- La longueur de la plus longue séquence de lettres minuscules * 2
- La longueur de la plus longue séquence de lettres majuscules * 2

Exemple :

Pour le mot de passe "**B@cSI_juin2015**", le score se calcule comme suit :

La somme des bonus = 14*4 + (14-3)*2 + (14-5)*3 + 6*5 = 135

Car { le nombre total de caractères = 14
le nombre de lettres majuscules = 3
le nombre de lettres minuscules = 5
le nombre de caractères non alphabétiques = 6

La somme des pénalités = 4*2 + 2*2 = 12

Car { la longueur de la plus longue séquence de lettres minuscules ("**juin**") = 4
la longueur de la plus longue séquence de lettres majuscules ("**SI**") = 2

Le score final = 135 – 12 = **123** ; puisque 123 ≥ 80 alors le mot de passe est considéré comme **"Très fort"**.

En disposant d'un fichier texte "**Motspass.txt**", situé sur la racine du disque **C**, dont chaque ligne contient un mot de passe, on se propose de :

- Générer un fichier d'enregistrements "**ForceMDP.dat**" où chaque enregistrement comporte le mot de passe lui-même, son score et sa force.
- Générer un fichier texte "**MDPfort.txt**" par la liste les mots de passe ayant une force égale à "**Très fort**" suivis de la liste des mots de passe ayant une force égale à "**Fort**" à raison d'un mot de passe par ligne, tout en séparant les deux listes par une ligne vide.

N.B : L'élève n'est pas appelé à remplir le fichier "**Motspass.txt**".

Travail demandé :

1- Analyser le problème en le décomposant en modules.

2- Analyser chacun des modules envisagés.

EXAMEN DU BACCALAURÉAT SESSION 2015

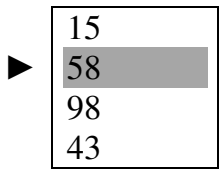
Corrigé Session principale

Épreuve : ALGORITHMIQUE ET PROGRAMMATION

Exercice 1 : *(10,5 points : 0,5 x 15 + 1 x 3)*

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

1) Soit un fichier d'entiers, ayant comme nom logique **F.** On suppose que le pointeur est positionné sur le deuxième entier comme indiqué ci-dessous.



N.B : Toutes les instructions données dans a), b) et c) sont valides.

a) Le contenu de X après l'exécution de l'instruction **Lire (F, X)** est :

F	15	V	58	F	98

b) L'instruction **Pointer (F, 3)** permet de positionner le pointeur sur l'enregistrement ayant la valeur :

F	58	F	98	V	43

c) Le contenu de Y après l'exécution de l'instruction **Y ← Fin_fichier (F)** est :

F	43	F	4	V	faux

2) Soit la suite U définie par : $\begin{cases} U_0 = 1 \\ U_n = 2*U_{n-1} + n \end{cases}$ avec n un entier supérieur ou égal à 1

a) U est une suite récurrente d'ordre :

V	1	F	2	F	5

b) Le 3ème terme de la suite U (U_2) est égal à :

F	5	V	8	F	9

c) L'algorithme permettant de calculer le $n^{ième}$ terme de la suite **U** (avec n > 1) est :

F

```
0) Def fn terme (n : entier) : entier
1) t[0] ← 1
2) Pour i de 1 à n faire
       t[i] ← 2*t[i-1]+n
   Fin pour
3) terme ← t[n]
4) Fin terme
```

V

```
0) Def fn terme (n : entier) : entier
1) Si n=0 alors terme ← 1
     Sinon
        terme ← 2*terme(n-1)+n
     Fin si
   Fin si
2) Fin terme
```

F

```
0) Def fn terme (n : entier) : entier
1) Up ← 1
2) Pour i de 2 à n faire
       Up ← 2*Up+i
   Fin pour
3) terme ← Up
4) Fin terme
```

Exercice 2 : (6 points)

```
DEF FN Verif_auto_nombre (N : entier) : Booléen
Résultat = Verif_auto_nombre ← n <> S+ i
        (S, i) = [i ← 0]
        Répéter
                i ← i + 1
                M ← i
                S ← 0
                Répéter
                        S ← S + M mod 10
                        M ← M div 10
                Jusqu'à (M =0)
        Jusqu'à (n = i + S) ou (i = n-1)
```

TDO	
Objet	Type
i	Entier
M	Entier
S	Entier

Exercice 3 : (6 points)

```
0) DEF FN Longueur_Max(epsilon, p:réel):réel
1) Lopt ← 0
   Smax ← 0
   Répéter
        Lopt ← Lopt+epsilon
        S ← Smax
        Smax ← Lopt*(p-2*Lopt)/Pi
   Jusqu'à ((Smax - S) <= 0) ou (Lopt >=p/2)
2) Longueur_Max ← Lopt - epsilon
3) Fin FN Longueur_Max
```

Problème : (17,5 points)

Analyse du programme principal :

Résultat = MDPF
MDPF =
    Associer (F_FORT,"c:\MDPFort.txt")
    Proc Generer_F_FORT (F_FORCE, F_FORT)
F_FORCE =
    Associer (F_FORCE,"c:\forceMDP.DAT")
    Proc Generer_F_FORCE (MP, F_FORCE)
MP =
    Associer (MP, "c:\Motspass.txt")

Tableau de déclaration de nouveaux types
ENRG = Enregistrement Mpass, Force : chaîne de caractères Score : entier Fin ENRG
F = Fichier de ENRG

Tableau de déclaration des objets globaux

Objet	Type/Nature
F_FORCE	F
F_FORT	Texte
MP	Texte
Generer_F_FORT	Procédure
Generer_F_FORCE	Procédure

Analyse de la procédure Generer_F_FORT:

```
Def Proc Generer_F_FORT (var F_FORCE : F; var F_FORT : Texte)
Résultat = F_FORT
F_FORT =
    [Ouvrir (F_FORCE), Récréer (F_FORT)]
    Tantque  non Fin_fichier (F_FORCE) Faire
        Lire (F_FORCE, e)
        Si e.force = "Très fort" Alors
                Ecrire_nl (F_FORT, e.Mpass)
        FinSI
    FinTantque
    Ecrire_nl (F_FORT)
    [Ouvrir (F_FORCE)]
    Tantque non Fin_fichier (F_FORCE) Faire
        Lire(F_FORCE, e)
        Si e.force = "Fort" Alors
                Ecrire_nl (F_FORT, e. Mpass)
        FinSI
    FinTantque
    Fermer (F_FORCE)
    Fermer (F_FORT)
```

T.D.O.L	
Objet	**Type/Nature**
e	ENRG

Analyse de la procédure Generer_F_FORCE :

```
Def Proc Generer_F_FORCE (var MP : Texte ; var F_FORCE : F)
Résultat = F_FORCE
F_FORCE = [Recréer (F_FORCE),    Ouvrir(MP)]
          Tantque non Fin_fichier (MP) Faire
              Lire_nl (MP, ch)
              r.Mpass ← ch
              Proc Nb_Min_Maj (ch, min, maj)
              r.score ← long (ch)*4 + (long (ch) – min) * 3 + (long (ch) - maj) * 2 + (long (ch) -
                    (min + maj)) * 5 – FN SeqMax (ch , "a", "z") * 2 – FN SeqMax (ch, "A", "Z") * 2
              Si r.score < 20 Alors r.force ←"Très faible"
              Sinon Si r.score < 40 Alors r.force ← "Faible"
              Sinon Si r.score < 60 Alors r.force ← "Moyen"
              Sinon Si r.score < 80 Alors r.force ← "Fort"
              Sinon r.force ← "Très fort"
              FinSi
              Ecrire (F_FORCE, r)
            FinTantque
         Fermer (F_FORCE)
         Fermer (MP)
```

T.D.O.L	
Objet	**Type/Nature**
Ch	Chaîne
r	ENRG
min	Entier
max	Entier
SeqMax	Fonction
Nb_Min_Maj	Procédure

Analyse de la procédure Nb_Min_Maj:

```
Def Proc Nb_Min_Maj (ch : Chaine ; Var min, maj : Entier)
Résultat = min, maj
(min, maj) = [ min←0 ; maj←0]
              Pour i de 1 à long (ch) Faire
               Si ch [i] dans ["a".."z"] Alors
                        min ← min+1
               FinSi
               Si ch [i] dans ["A".."Z"] Alors
                        maj ← maj+1
               FinSi
             FinPour
```

T.D.O.L	
Objet	**Type/Nature**
i	Entier

Analyse de la fonction SeqMax:

```
Def FN SeqMax (ch : Chaine ; c1, c2 : Caractère) : Entier
Résultat = SeqMax ← Min
Min = [ncour ←0; npred ←0]
       Pour i de 1 à long (ch) Faire
         Si ch[i] dans [c1..c2] Alors
              ncour ← ncour+1
        Sinon
              Si npred < ncour Alors
                    npred ← ncour
              FinSi
              Ncour ← 0
        FinSI
      FinPour
      Si npred < ncour Alors  Min ← ncour
      Sinon  Min ← npred
      FinSi
```

T.D.O.L	
Objet	**Type/Nature**
Min, ncour, npred, i	Entier

RÉPUBLIQUE TUNISIENNE MINISTÈRE DE L'ÉDUCATION ⇘⇘⇘⇘ EXAMEN DU BACCALAURÉAT SESSION 2015	Épreuve : **ALGORITHMIQUE ET PROGRAMMATION** Durée : 3 h Coefficient : 2,25
Section : **Sciences de l'informatique**	**Session de contrôle**

Le sujet comporte 4 pages numérotées de 1/4 à 4/4
Les réponses à l'exercice 1 doivent être rédigées sur les pages 1/4 et 2/4
qui doivent être remises avec la copie

Exercice 1 (3 points)

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

1- Un module est appelé dans le corps de sa propre définition.

- ☐ Jamais
- ☐ Possible
- ☐ Toujours

2- Le module ci-dessous calcule le factoriel de tout entier supérieur ou égal à zéro.

☐
```
0)  Def FN Fact (m : Entier) : Entier Long
    1) Si m □ 2 Alors Fact ← 1
       Sinon Fact ← m* FN Fact(m-1)
       FinSi
    2) Fin Fact
```

☐
```
0)  Def FN Fact (m : Entier) : Entier Long
    1) Si m = 1 Alors Fact ← 1 Sinon
       Fact ← m * FN Fact(m-1)
       FinSi
    2) Fin Fact
```

☐
```
0)  Def FN Fact (m : Entier) : Entier Long
    1) Si m = 0 Alors Fact ← 1 Sinon
       Fact ← m * FN Fact(m)
       FinSi
    2) Fin Fact
```

3- Pour résoudre le problème des tours de Hanoï à 3 disques, il faut réaliser le nombre minimal de mouvements de disques suivant :

- ☐ 3
- ☐ 6
- ☐ 7

4- Lorsque la condition d'arrêt d'un module récursif est vérifiée,

- ☐ le module ne fera plus d'appels à lui-même.
- ☐ le module renvoie une erreur.
- ☐ le module arrête l'exécution du programme.

Exercice 2 (4 points)

On se propose de calculer une valeur approchée de π, selon la méthode décrite ci-dessous : 1)

On remplit une matrice M de la façon suivante :

- ✓ M[0,0] = 1
- ✓ M[L,C] = la somme des C derniers éléments de la ligne (L-1) avec L □ 0 et 0 ≤ C ≤ L

2) On calcule pour chaque ligne L le résultat $R_L = \frac{* * \lfloor \quad , \quad \rfloor}{[\ , \ ]}$

Ce traitement s'arrête lorsque la différence entre R_L et R_{L-1} est inférieure ou égale à □ (avec $10^{-4} \leq \square \leq 10^{-1}$) et par conséquent la valeur approchée de π sera égale à R_L.

Exemple :

Pour □ = 10^{-3} et en procédant au remplissage de la matrice M ligne par ligne, on obtient le contenu suivant:

	0	1	2	3	4	5	6	7	8	9
0	1									
1	0	1								
2	0	1	1							
3	0	1	2	2						
4	0	2	4	5	5					
5	0	5	10	14	16	16				
6	0	16	32	46	56	61	61			
7	0	61	122	178	224	256	272	272		
8	0	272	544	800	1024	1202	1324	1385	1385	
9	0	1385	2770	4094	5296	6320	7120	7664	7936	7936

- Le contenu de la case M[6,3] est obtenu en calculant la somme des 3 derniers éléments de la ligne 5 (14+16+16= 46).
- Le contenu de la case M[9,6] est obtenu en calculant la somme des 6 derniers éléments de la ligne 8 (800+1024+1202+1324+1385+1385 = 7120).

Le calcul s'arrête à la ligne 9 car:

- A la ligne n° 8, $R_8 = \frac{* * [\ , \ ]}{[\ , \ ]} = \frac{* *}{\quad} = 3.142238267\ldots$
- A la ligne n° 9, $R_9 = \frac{* * [\ , \ ]}{[\ , \ ]} = \frac{* *}{\quad} = 3.141381048\ldots$

La différence entre R_9 et R_8 est égale à 0,000857219 = 0,857219*10^{-3} qui est inférieure à □ (10^{-3}), par conséquent la valeur approchée de π est R_9 = 3.141381048… Travail demandé :

Ecrire une analyse d'un module intitulé "Calcul_Pi" qui permet de calculer, à □ près, une valeur approchée de π, en utilisant la méthode décrite ci-dessus, sachant que □ est déjà saisi dans l'analyse du programme principal.

Exercice 3 (3,5 points)

En mathématiques, la suite b_n de Baum-Sweet (avec $n \geq 0$) est une suite dont les termes valent 0 ou 1. Elle est définie par : $b_n = 0$, si la représentation binaire de n contient au moins un bloc composé d'un nombre impair de 0

$b_n = 1$, sinon.

Exemples :

- $b_4 = 1$ car la représentation binaire de 4 est 100, qui ne contient aucun bloc de nombre impair de 0.
- $b_{68} = 0$ car la représentation binaire de 68 est 1000100, qui contient un bloc formé d'un nombre impair de 0 (bloc de 3 zéros successifs).
- $b_{261} = 0$ car la représentation binaire de 261 est 100000101, qui contient au moins un bloc formé d'un nombre impair de 0 (bloc de 5 zéros successifs).

Travail demandé :

Ecrire un algorithme d'un module qui permet d'afficher les P premiers termes de la suite de Baum-Sweet, sachant que P est un entier strictement positif déjà saisi dans l'algorithme principal.

Problème (9,5 points)

En utilisant un ordinateur, même ayant un seul processeur, nous remarquons qu'on peut lancer plusieurs programmes en même temps. Or, nous savons qu'un seul processeur ne peut exécuter qu'un seul programme à la fois. Cette notion de "multitâche" est obtenue grâce au système d'exploitation.

Pour ce faire, le système d'exploitation utilise une technique appelée ordonnancement des processus, qui consiste à gérer l'allocation des différents processus au processeur.

Il existe plusieurs algorithmes d'ordonnancement des processus, tels que :

- FIFO (First In First Out) : Le processus qui arrive le premier sera le premier à être exécuté.
- LIFO (Last In First Out) : Le processus qui arrive le dernier sera le premier à être exécuté.
- SJF (Shortest Job First) : Le processus qui a une durée d'exécution minimale sera le premier à être exécuté.

On se propose d'élaborer un nouvel ordonnancement basé sur les deux méthodes : FIFO et SJF, comme expliqué ci-dessous :

1) Remplir un fichier d'enregistrements intitulé "Processus.dat", situé sur la racine du disque C, par N processus prêts à être exécutés (avec $3 \leq N \leq 200$), sachant qu'un processus est caractérisé par :
   - un code, qui est une chaîne de caractères formée par la lettre "P" suivie d'un nombre qui commence de 1 et s'incrémente automatiquement de 1 pour chaque nouveau processus (P1, P2, …, P100,….).
   - une durée d'exécution exprimée en millisecondes.

   NB : L'ordre de remplissage des processus dans le fichier "Processus.dat" représente l'ordre d'ordonnancement FIFO.
2) A partir du fichier "Processus.dat", appliquer l'algorithme d'ordonnancement SJF pour classer les processus dans un nouveau fichier d'enregistrements intitulé "Ord_SJF.dat".
3) A partir des fichiers "Processus.dat" et "Ord_SJF.dat", générer un fichier texte intitulé "Ord_Nouv.txt", contenant les codes des processus, chacun sur une ligne, et ce de la manière suivante :
   a. Commencer par placer chaque processus ayant le même rang dans les deux fichiers "Processus.dat" et "Ord_SJF.dat".

b. Ensuite, placer le reste des processus selon leur ordre d'apparition dans le fichier "Ord_SJF.dat".

Exemple :

- Pour le fichier "Processus.dat" suivant :

Code	Durée
P1	3
P2	1
P3	2
P4	3
P5	1
P6	5

- En appliquant l'algorithme d'ordonnancement SJF, on obtient le fichier "Ord_SJF.dat" suivant :

Code	Durée
P2	1
P5	1
P3	2
P1	3
P4	3
P6	5

- le fichier "Ord_Nouv.txt" généré sera le suivant :

P3
P6
P2
P5
P1
P4

Travail demandé :

1- Analyser le problème en le décomposant en modules.
2- Analyser chacun des modules envisagés.

RÉPUBLIQUE TUNISIENNE MINISTÈRE DE L'ÉDUCATION ◈◈◈◈ EXAMEN DU BACCALAURÉAT SESSION DE JUIN 2015	Epreuve : ALGORITHMIQUE ET PROGRAMMATION **Corrigé**
Section : Sciences de l'informatique	Session de contrôle

Exercice 1 : (6 points = 0,5x12)

1- Un module est appelé dans le corps de sa propre définition.

[F] Jamais

[V] Possible

[F] Toujours

2- Le module suivant calcule le factoriel de tout entier supérieur ou égal à zéro.

[V] 0) Def FN Fact (m : Entier) : Entier Long
1) Si m < 2 Alors Fact ← 1
Sinon Fact ← m* FN Fact(m-1)
FinSi
2) Fin Fact

[F] 0) Def FN Fact (m : Entier) : Entier Long
1) Si m = 1 Alors Fact ← 1
Sinon Fact ← m * FN Fact(m-1)
FinSi
2) Fin Fact

[F] 0) Def FN Fact (m : Entier) : Entier Long
1) Si m = 0 Alors Fact ← 1
Sinon Fact ← m * FN Fact(m)
FinSi
2) Fin Fact

3- Pour résoudre le problème des tours de Hanoï à 3 disques, il faut réaliser le nombre minimal de mouvements de disques suivant :

[F] 3

[F] 6

[V] 7

4- Lorsque la condition d'arrêt d'un module récursif est vérifiée,

[V] le module ne fera plus d'appels à lui-même.

[F] le module renvoie une erreur.

[F] le module arrête l'exécution du programme.

Exercice 2 : (8 points)

```
DEF FN Calcul_Pi (E : réel) : réel
Résultat = Calcul_Pi ← Pii
Pii =   [M[0,0]← 1 , i← 0 , Pii ← 0]
        Répéter
                i← i+1
```

```
                P ← Pii
                Pour j de 0 à i faire
                    S ← 0
                    Pour k de i-1 à i-j faire
                        S ← S + M[i-1,k]
                    Fin Pour
                    M[i,j] ← S
                Fin Pour
                Pii ← 2*i*M[i-1,i-1]/M[i,i]
    Jusqu'à Abs (P-Pii) ≤ E
Fin Calcul_Pi
```

Tableau de déclaration des objets locaux

Objet	Type
Pii, P, S	Réel
M	Tableau de 20x20 de réels
i, j, k	Entier

Exercice 3 (7 points)

```
0) DEF PROC Baum_Sweet (P : Entier)
1) Pour  i de 0 à P-1 Faire
     Ecrire(Verif(i))
   FinPour
2) Fin Baum_Sweet

0) DEF FN Verif (i : Entier) : Entier
1) ch←Conv_2 (i)
   j←0
    Répéter
            j←j+1
            n←0
            Tant que (ch[j]= "0") et( j<= Long (ch)) Faire
                j←j+1
                n←n+1
            FinTantque
     Jusqu'à (n mod 2 = 1) ou (j = Long(ch))
2) verif← 1- n mod 2
3) Fin Verif

0) DEF FN Conv_2 (i : Entier) : Chaîne
1) ch←""
   Répéter
     ch← chr(48 + i mod 2) + ch
     i←i div 2
   Jusqu'à i=0
2) conv_2← ch
3) Fin Conv_2
```

Problème : (19 points)

1) Analyse du PP

Résultat = Nouv
Nouv = [Associer (Nouv, 'C:\Ord_Nouv.txt')] PROC Nouv_Ord (Fifo, F_SJF, N, Nouv)
F_SJF = [Associer (F_SJF, 'C:\Ord_SJF.dat')] PROC SJF (Fifo, F_SJF, N)
Fifo= [Associer (Fifo, 'C:\Processus.dat')] PROC Remplir_F (Fifo, N)
N= PROC Saisir (N)

Tableau de déclarations de nouveaux types

Types
Enr_Process = Enregistrement Code : Chaîne[10] Duree : Entier Fin Enr_Process F_Process = Fichier de Enr_Process

Tableau de déclaration des objets globaux

Objet	Type
Fifo, F_SJF	F_Process
Nouv	Text
N	Entier
Saisir, Remplir_F, SJF, Nouv_Ord	Procédure

2)

```
DEFPROC Saisir (Var N : Entier)
Résultat =
    Répéter
        N = Donnée ("Donner N : ")
    Jusqu'à N Dans [3..200]
Fin Saisir
```

```
DEFPROC Remplir_F (Var F : F_Process; N : Entier)
Résultat = F
F = [Recréer (F)]
    Pour i de 1 à N Faire
        Process.duree = Donnée ("Entrer la durée du processus n° ",i, " : ")
        Convch (i, chi)
        Process.code ← "P"+Chi
        Ecrire (F, Process)
    FinPour
    Fermer (F)
Fin Remplir_F
```

Tableau de déclarations des objets locaux

Objet	Type
Process	Enr_Process
i	Entier
chi	Chaîne

```
DEFPROC SJF (Var F, F_SJF : F_Process; N : Entier)
Résultat = F_SJF
F_SJF = [Recréer (F_SJF)]
                Pour i de 1 à N Faire
                        Ecrire (F_SJF, T[i])
                FinPour
                Fermer (F_SJF)
T = [Ouvrir (F)]
        Pour i de 1 à N Faire
                Lire (F, Process)
                T[i] ← Process
        FinPour
        Tri_SJF (T, N)
        Fermer (F)
Fin SJF
```

Tableau de déclarations de nouveaux types locaux

Types
Tab = Tableau de 200 Enr_Process

Tableau de déclarations des objets locaux

Objet	Type
i	Entier
Process	Enr_Process
T	Tab
Tri_SJF	Procédure

```
DEFPROC Tri_SJF (Var T : Tab; M : Entier)
Résultat = T
T = []
        Pour k de 2 à n Faire
                j ← k
                Tantque (T[j-1].duree > T[j].duree) et (j>1) Faire
                        aux ←T[j]
                        T[j] ← T[j-1]
                        T[j-1] ← aux
                        j ← j-1
                FinTantque
      FinPour
```

Tableau de déclarations des objets locaux

Objet	Type
j, k	Entier
aux	Enr_Process

```
DEFPROC Nouv_Ord (Var F1, F2 : F_Process; N : Entier; Var F : Text)
Résultat = F
F = [Ouvrir (F1); Ouvrir (F2); Recréer (F)]
        Pour i de 1 à N Faire
                Lire (F1, Process1)
                Lire (F2, Process2)
                Si Process1.Code = Process2.Code Alors
                        Ecrire_nl (F, Process2.Code)
                FinSi
```

```
        FinPour
        Ouvrir (F1)
        Ouvrir (F2)
        Pour i de 1 à N Faire
                Lire (F1, Process1)
                Lire (F2, Process2)
                Si Process1.Code ≠ Process2.Code Alors
                        Ecrire_nl (F, Process2.Code)
                FinSi
        FinPour
    Fermer (F1)
    Fermer (F2)
    Fermer(F)
Fin Nouv_Ord
```

Tableau de déclarations des objets locaux

Objet	Type
Process1, Process2	Enr_Process
i	Entier