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 $0 \leq C \leq L$

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 $10^{-4} \leq$ □ $\leq 10^{-1}$) 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.