

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

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

- ☐ Jamais
☒ Possible
☐ Toujours

2- Le module suivant 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 : (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

