

EXAMEN DU BACCALAUREAT 2013

Section : Sc. Info.

Epreuve : Algorithmique et Programmation

CORRIGE**Exercice 1 (2,5 points = 0,25 + 2 + 0,25)**1- Déterminer le type de retour de la fonction **Inconnue**.*Entier*2- Donner la trace d'exécution ainsi que les résultats retournés par la fonction **Inconnue** pour les valeurs de A et B suivantes :

- A = 6 et B = 15

	<i>Inconnue</i>
A < B	<i>Inconnue</i> (6, 9) * 15 Div 9
A < B	<i>Inconnue</i> (6, 3) * 9 Div 3
A > B	<i>Inconnue</i> (3, 3) * 6 Div 3
A = B	3

Le résultat retourné par la fonction est égal à $3 * 6 \text{ Div } 3 * 9 \text{ Div } 3 * 15 \text{ Div } 9 = 30$

- Pour A = 8 et B = 3

	<i>Inconnue</i>
A > B	<i>Inconnue</i> (5, 3) * 8 Div 5
A > B	<i>Inconnue</i> (2, 3) * 5 Div 2
A < B	<i>Inconnue</i> (2, 1) * 3 Div 1
A > B	<i>Inconnue</i> (1, 1) * 2 Div 1
A = B	1

Le résultat retourné par la fonction est égal à $1 * 2 \text{ Div } 1 * 3 \text{ Div } 1 * 5 \text{ Div } 2 * 8 \text{ Div } 5 = 24$ 3- Dédurre le rôle de la fonction **Inconnue**.La fonction *Inconnue* retourne le **PPCM** de deux entiers A et B**Exercice 2 (4 points = 1 + 3)**

1- Donner la décomposition en une somme de puissances de 2 distinctes pour les nombres 31 et 56.

- $31 = 2^0 + 2^1 + 2^2 + 2^3 + 2^4$
- $56 = 2^3 + 2^4 + 2^5$

2- Ecrire un algorithme d'un module qui permet d'afficher la décomposition d'un entier N, en une somme de puissances de 2 distinctes.

0) DEF PROC Affiche_puissance (N : entier)

1) P ← 0

Ch ← ""

Tantque (N > 0) Faire

Si N mod 2 = 1

Alors convch(P, PP)

Ch ← Ch + "2^" + PP + "+"

Fin si

P ← P + 1

N ← N Div 2

FinTantque

2) Efface(Ch, long(Ch), 1)

3) Ecrire(Ch)

4) Fin Affiche_puissance

Exercice 3 (3,5 points = 1.25 + 2,25)

1- En utilisant la définition donnée ci-dessus, écrire un algorithme d'une fonction nommée

Combinaison permettant de calculer C_n^p .

Algorithme de la fonction combinaison :

0) DEF FN Combinaison (n, p : entier) : Entier long

1) Si (p = 0) ou (n = p) Alors Combinaison ← 1

 Sinon Combinaison ← Combinaison(n-1,p) + Combinaison(n-1,p-1)

 FinSi

2) Fin FN Combinaison

2- Utiliser la fonction **Combinaison** afin d'écrire un algorithme d'un module qui permet de déterminer une valeur approchée de S à **epsilon** près.

Algorithme de la fonction Approchée_S :

0) DEF FN Approchée_S (epsilon : réel) : Réel

1) S ← 1

 I ← 0

 puis ← 1

 Répéter

 I ← i+1

 S_pred ← S

 puis ← puis * 2

 S ← S + puis * 1/((2 * i+1) * FN combinaison(2*i,i))

 Jusqu'à ABS (S_pred-S) ≤ epsilon

2) Approchée_S ← S

3) Fin Approchée_S

Problème (10 points)

1- Analyse du programme principal :

Résultat = fc

fc = Associer(fc, "c:\fcode.txt")

Proc Former_fcode(fc, fint, N)

fint = Associer(fint, "c:\fint.txt")

Proc Former_finter(fint, M, N)

(M, N) = Proc Remplissage(M, N)

T.D.N.T.

Type
Tab = Tableau de 20 x 20 de caractères

T.D.O.G

<i>Objet</i>	<i>Type</i>
<i>fc</i>	<i>Fichier texte</i>
<i>fint</i>	<i>Fichier texte</i>
<i>M</i>	<i>Tab</i>
<i>N</i>	<i>Entier</i>
<i>Remplissage</i>	<i>Procédure</i>
<i>Former_finter</i>	<i>Procédure</i>
<i>Former_fcode</i>	<i>Procédure</i>

2- Analyse des modules :

Analyse de la procédure Remplissage :

DEF PROC Remplissage(Var M : Tab ; Var N :Entier)

Résultat = M,N

M = [N= donnée]

Pour i de 1 à N faire

Pour j de 1 à N faire

M[i,j] ← CHR(Aléatoire(26) +65)

FinPour

FinPour

Fin Remplissage

Analyse de la procédure Former_finter :

DEF PROC Former_finter(Var finter : text, M : Tab, N : Entier)

Résultat = finter

finter = [i ← 1, j ← N]

Répéter

Ch ← ""

Pour k de i à j Faire

Ch ← Ch + M[i,k]

FinPour

Pour K de i+1 à j Faire

Ch ← Ch + M[k,j]

FinPour

Pour k de j-1 à i Pas -1 Faire

Ch ← Ch + M[j,k]

Fin Pour

Pour k de j-1 à i+1 Pas -1 Faire

Ch ← Ch + M[k,i]

FinPour

Ecrire_nl(Finter, Ch)

i ← i+1

j ← j-1

Jusqu'à i > j

Fin Former_finter

T.D.O.L

<i>Objet</i>	<i>Type</i>
<i>i</i>	<i>Entier</i>
<i>J</i>	<i>Entier</i>
<i>k</i>	<i>Entier</i>
<i>Ch</i>	<i>Chaîne</i>

Analyse de la procédure Former_fcode :

```
DEF PROC Former_fcode(Var fcode , finter : text)
  Résultat = fcode
  fcode = [Ouvrir(fcode),Ouvrir(finter), Ecrire_nl(fcode,FN Conv_base(n,2))]
  Tantque Non(FinFichier(finter)) faire
    Lire_nl(finter,ch1)
    Ch2 ← ""
    Pour i de 1 à Long(ch1) faire
      ch2 ← ch2 + FN Conv_base(Ord(ch1[i]),16) + "#"
    FinPour
    Ecrire_nl(fcode,ch2)
  FinTantque
  Fermer(fcode), fermer(finter)
Fin Former_fcode
```

T.D.O.L

<i>Objet</i>	<i>Type</i>
<i>Conv_base</i>	<i>fonction</i>
<i>i</i>	<i>Entier</i>
<i>ch1</i>	<i>Chaîne</i>
<i>ch2</i>	<i>Chaîne</i>

Analyse de la fonction Conv_base :

```
DEF FN Conv_base(N,B: Entier) : Chaîne
  Résultat = Conv_base ← Ch
  Ch = [Ch ← ""]
  Répéter
    R ← N Mod B
    Si R ≥ 10 Alors Ch_R ← Chr(55+R)
    Sinon Convch (R, Ch_R)
  FinSi
  Ch ← Ch_R + Ch
  N ← N Div B
  Jusqu'à N = 0
Fin Conv_base
```

T.D.O.L.

<i>Objet</i>	<i>Type</i>
<i>Ch, Ch_R</i>	<i>Chaîne</i>
<i>R</i>	<i>Entier</i>