Correction Algorithmique et programmation
Session principale 2009

Exercice 1 (4 pts, on acceptera toute réponse équivalente)

a) L'algorithme proposé est celui de la méthode du tri par insertion. **(1 point)**

b) La méthode consiste à parcourir le tableau à trier à partir du 2ème élément et essaye de placer l'élément en cours à sa bonne place parmi les précédents. De cette façon, la partie dans laquelle on cherche à placer l'élément en cours reste toujours triée. L'opération de placement peut nécessiter le décalage d'un bloc d'un pas vers l'avant. **(1,5 point)**

c) Algorithme de la procédure DECALER **(1,5 point = 0,5 pour le mode de passage + 1 pour le corps de la procédure)**

```
0) Procédure DECALER(i: entier, VAR j :  Entier, VAR T : W)
1) j← i
2) Tant Que (T[j-1]>aux) et (j > 1) Faire
        T[j] ← T[j-1]
        j← j-1
   Fin Tant Que
3) Fin DECALER
```

NB:

- **Il est possible de remplacer la structure Tant que par répéter.**
- **On acceptera toute réponse équivalente**

Exercice 2 :

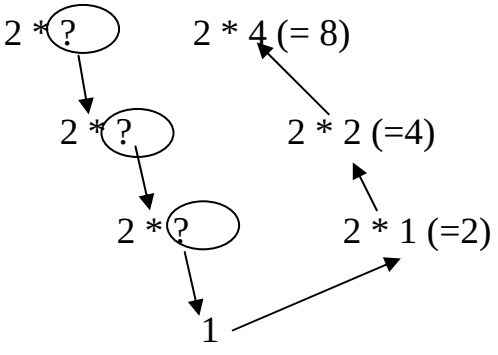
1°) Ordre de récurrence de la fonction **inconnu1** = 1
Ordre de récurrence de la fonction **inconnu2** = 2

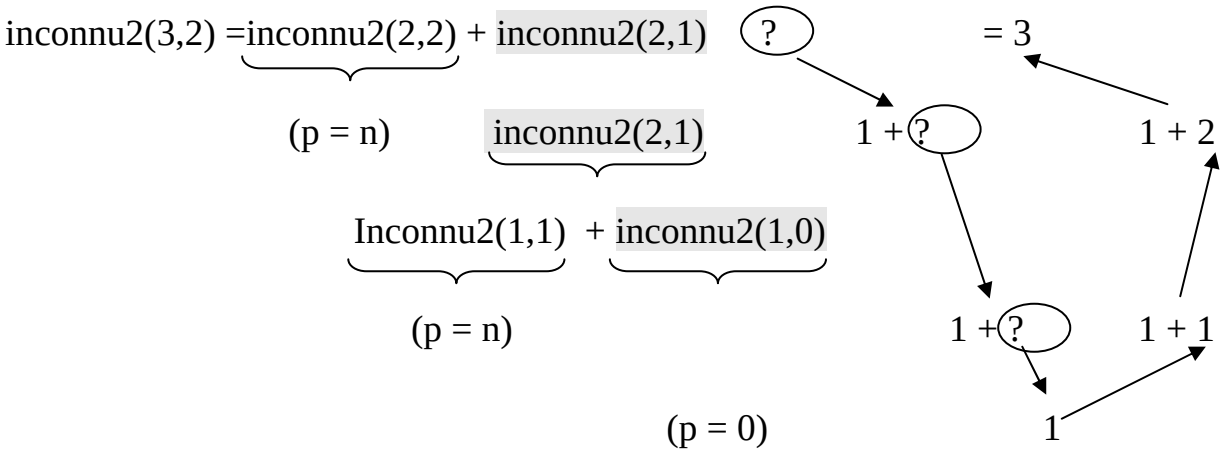
2°) inconnu1(2,3) = 2 * **inconnu1(2,2)**

2 * inconnu1(2,1)

2 * inconnu1(2,0)

(n = 0)





3°) La fonction inconnu1 permet de calculer x^n
La fonction inconnu2 permet de calculer C_n^p

4°) Le développement de $(x+1)^n$ est égal à $\sum_{p=0}^{n} C_n^p \; x^p$

1) DEF PROC develop (n : entier)
2) [STR(n,chn) ; dev←"x^0"]
   Pour p de 1 à n Faire
      STR(inconnu2(n, p),Cnp)
      STR(p,chp)
      dev ← dev + " + " + Cnp + "." + " x^" + chp
   FinPour
3) Ecrire(dev)
4) Fin develop

NB: On acceptera toute réponse équivalente

Barème			
Question n°1	Question n°2	Question n°3	Question n°4
1pt=0,5+0,5	1pt= 0,5+0,5 Trace=0,25 Résultat= 0,25	1pt = 2*0,5	1 pt

Problème :

Analyse (8 Pts)	**Algorithme** • ***Pour être évalué, l'algorithme doit présenter un contenu en***

	relation avec le problème demandé
1) <u>PP (1,5 pt)</u> • **Cohérence=0,5** • **Modularité= 1** **2) <u>Saisie (1 pt=4*0,25)</u>** **3) <u>Remplissage (2 pts)</u>** • **<u>Parcours (1 pt)</u>** • **<u>Détermination de la valeur (1 pt)</u>** **<u>4) Recherche des chemins (0,75 pt)</u>** **<u>5) Remplissage du fichier (0,75 pt)</u>** **<u>6) Affichage (1 pt)</u>** **7) Divers (TDO+mode de passage des paramètres…) (1 pt)**	**<u>PP (0,5 pt)</u>** **<u>Saisie (0,5 pt)</u>** **Remplissage (1,5 pt)** **<u>Traitement (1,5 pt)</u>**

1) Analyse du programme principal nommé **Les_chemins**

Résultat = affichage , fichier_chemins
affichage , fichier_chemins = PROC Traitement (p,q,M)
(p,q)= PROC Saisie(p,q)
M= PROC Remplissage(p,q,M)

Tableau de déclaration des nouveaux types

Type
Matrice = tableau de 1..32 , 1..32 entier

Tableau de déclaration des objets globaux

Objet	Type/Nature	Rôle
p	entier	Nombre de lignes de la matrice
q	entier	Nombre de colonnes de la matrice
M	Matrice	Matrice à remplir
Traitement	procédure	Permet d'afficher les n° des chemins et de remplir le fichier chemin
Saisie	procédure	Permet de saisir p et q
Remplissage	procédure	Permet de remplir la matrice

Analyse de la Procédure saisie

```
DEF PROC saisie (VAR n,m : entier)
Résultat = n,m
(n,m) = [       ] Répéter
                    n=donnée ("Entrer le nombre de lignes : ")
                    m=donnée ("Entrer le nombre de colonnes: ")
                Jusqu'à (5 ≤ n) et (n ≤ m) et (m ≤ 32)
Fin Saisie
```

Analyse de la Procédure **remplissage**

```
DEF PROC remplissage
(n,m : entier ; VAR A : Matrice)
Résultat = A
A = [      ] Pour i de 1 à n Faire
           [    ] Pour j de 1 à m Faire
                  [A[i,j] ← 0, L ← i, C ← j]

                  Répéter

                  [] Si (L MOD 2) + (C MOD 2)=2
                     alors   A[i, j]← 1
                     sinon
                              L ← L DIV 2
                              C ← C DIV 2
                      Fin Si
                  jusqu'à (L*C =0) ou (A[i,j]=1)
               Fin Pour
           Fin Pour
Fin Remplissage
```

Commentaires

- Par défaut, on affecte **0** à la case **A[i,j]**. La valeur de cette case sera modifiée si ses indices (**i** et **j**) vérifient la condition de l'énoncé.
- Début de la vérification :

1. Conversion en binaire de i et j et test de la condition:
* Division de **i** et **j** par **2**.
* **Si** pour cette division, les **2** restes valent **1**, alors leur **somme** (1+1) vaut **2**, donc, selon l'énoncé, on affecte **1** à **A[i,j].**
Sinon, on poursuit la conversion

2. la vérification s'arrête à la fin de la conversion (au moins, un des quotients s'annule (L*C=0)) ou bien si la condition de l'énoncé est vérifiée (A[i,j]=1)

Tableau de déclaration des objets la procédure remplissage

Objet	Type/Nature	Rôle
i	entier	Compteur
j	entier	Compteur
L	entier	Variable intermédiaire pour vérifier que i et j ont un 1 en commun sur la même position
C	entier	Variable intermédiaire pour vérifier que i et j ont un 1 en commun sur la même position

Analyse de la Procédure **Traitement**

```
DEF PROC Traitement
(n,m : entier, A : Matrice)
Résultat =affichage, chemin
(affichage, chemin) = [nbchemin ← 0 ;
Associer (chemin,"D:\chemins.txt")
recréer (chemin) ; Ecrire_nl (chemin,n," ",m) ]
      Pour i de 1à m faire
            Si Fn Verif_Chemin (n,i,A) Alors
                écrire(i)
                 nbchemin ← nbchemin + 1
                  Ecrire_nl (chemin,i)
             Fin si
       Fin pour
       Ecrire_nl (chemin, nbchemin)
       Fermer (chemin)
Fin Traitement
```

Commentaires

- Traitement d'initialisation
- Traitement de toutes les colonnes (**de 1 à m**)
- Affichage du n° de la colonne (**i**) qui vérifie les **conditions** et **incrémentation** du compteur de chemins valides (**nbchemin**)
- Sauvegarde des résultats dans le fichier **chemin**

Tableau de déclaration des objets de la procédure Traitement

Objet	Type/Nature	Rôle
i	entier	Compteur
nbchemin	entier	Nombre de chemins
chemin	texte	Fichier texte qui va contenir le nombre de lignes, le nombre de colonnes, le numéro de chaque chemin et le nombre de chemins
Verif_chemin	Fonction	Permet de vérifier si une colonne constitue un chemin

Analyse de la Fonction **Verif_Chemin**

	DEF FN Verif_chemin (n,k : entier ; M : Matrice) : booléen	
S	LDE	OU
2 1 3	Résultat = Verif_chemin ← vérif vérif=[vérif←vrai ; i←0] répéter i ← i + 1 Si (M[i,k] = 0) et (M[i+1,K]=0) Alors vérif ← faux Finsi Jusqu'à (vérif = faux) ou (i = n-1) Fin Verif_chemin	i verif

Commentaires

- Le chemin (**une colonne k**) est valide s'il ne contient pas deux **0 consécutifs (verticalement)**.
- Le traitement s'arrête si le chemin k n'est pas valide (**vérif=faux**) ou si l'avant dernière ligne a été traitée **i=n-1**, cas d'un chemin valide).

Tableau de déclaration des objets la fonction Verif_chemin

Objet	Type/Nature	Rôle
i	entier	Compteur
vérif	booléen	Permet de vérifier si une colonne constitue un chemin

2) Les algorithmes se déduisent directement des analyses correspondantes.

Algorithme du PP

0) Début Les_chemins
1) PROC Saisie(p,q)
2) PROC Remplir(p,q,M)
3) PROC Traitement (p,q,M)
4) Fin Les_chemin

Algorithme de la procédure Saisie

0) DEF PROC saisie (VAR n,m : entier)
1) Répéter
   Ecrire ("Entrer le nombre de lignes : ") ; lire(n)
   Ecrire ("Entrer le nombre de colonnes : ") ; lire(m)
   Jusqu'à (5≤n) et (n≤m) et (m≤32)
2) Fin chemin

Algorithme de la procédure Remplissage

0) DEF PROC remplissage(n,m : entier ; VAR A : Matrice)
1) Pour i de 1 à n Faire
   Pour j de 1 à m Faire
   A[i,j] ← 0 ; L ← i ; C ← j
   Répéter
   Si (L MOD 2) + (C MOD 2)=2 alors A[i, j] ← 1
   Sinon
   L ← L DIV 2
   C ← C DIV 2
   Fin Si
   Jusqu'à L*C =0 ou (A[i,j]=1)
   Fin Pour
   Fin Pour
2) Fin remplissage

Algorithme de la procédure Traitement

0) DEF PROC Traitement (n,m : entier, A : Matrice)
1) nbchemin ← 0
   Associer(chemin,"D:\chemins.txt")
   recréer(chemin)

```
Ecrire_nl(chemin,n," ",m)
Pour i de 1à m faire
        Si Fn Verif_Chemin(n,i,A)
                    Alors
                     écrire(i)
                     nbchemin ← nbchemin + 1
                    Ecrire_nl(chemin,i)
                Finsi
 Fin pour
 Ecrire_nl(chemin,nbchemin)
```

2) Fermer(chemin)
3) Fin Traitement

Algorithme de la fonction Verif_Chemin

0) DEF FN Verif_chemin(n,k : entier ; M : Matrice) : booléen
1) vérif←vrai ; i←0
   répéter
      i ← i + 1
      Si (M[i,k] = 0) et (M[i+1,K]=0) Alors vérif ← faux Finsi
   Jusqu'à (vérif = faux) ou (i = n-1)
   Verif_chemin ← vérif
2) Fin Verif_chemin