

Proposé par : Mr. Zied Fridhi

Lycée Manzel Hayet
Monastir

4^{ème} Sciences de l'Informatique

Durée : 2 Heures

04 Décembre 2013

Devoir de Synthèse n°1

Algorithmique et Programmation

Exercice n°1 : (3 points)

Soit l'algorithme de la fonction suivante :

```
0) DEF FN Inconnue (St : chaîne ) : chaîne
1) Si St = "" alors Inconnue ← ""
   Sinon Inconnue ← St[long(St)] + FN Inconnue(sous-
   chaîne(St,1,long(St)-1))
   Fin si
2) Fin Inconnue
```

Q1 : Exécuter manuellement l'algorithme de la fonction Inconnue avec :

▲ St = "RADAR" | ▲ St = "algo"

Q2 : Quel est le rôle de cette fonction.

Exercice n°2 : (5 points)

Une coupe C[i..j] d'un tableau T est une suite d'éléments consécutifs T[i],T[i+1],... T[j-1],T[j] ou $1 \leq i \leq j \leq n$.

Cette coupe est un palindrome si elle est égale à la coupe obtenue en prenant ses éléments à l'envers.

Exemple : pour le tableau T suivant

14	7	-9	8	5	1	5	8	9	25	7
----	---	----	---	---	---	---	---	---	----	---

La coupe C[4..8] est une coupe palindrome.

Q1 : Ecrire l'algorithme d'une fonction récursive qui vérifie si une coupe de T est palindrome ou non.

Q2 : Ecrire l'algorithme d'une procédure qui permet de transférer les éléments d'une coupe palindrome d'un tableau T vers un fichier typé "fichpal.dat" que vous créerez.

Exercice n°3 : (12 points)

Deux chevaliers vont jouer à tour de rôle sur un damier. Le principe du jeu est le suivant :

Chaque chevalier joue sur une matrice carrée de taille (n*n) en se déplaçant sur les carreaux de sa matrice. Chacun va partir de la position initiale devant la première case de sa matrice(en dehors de la matrice) pour arriver à la dernière case.

Le chevalier gagnant est celui qui arrive le premier à la dernière case de sa matrice.

Les cases des deux matrices sont initialisées à zéro. Après déplacement, la case dans laquelle s'arrête un chevalier est modifiée à 1.

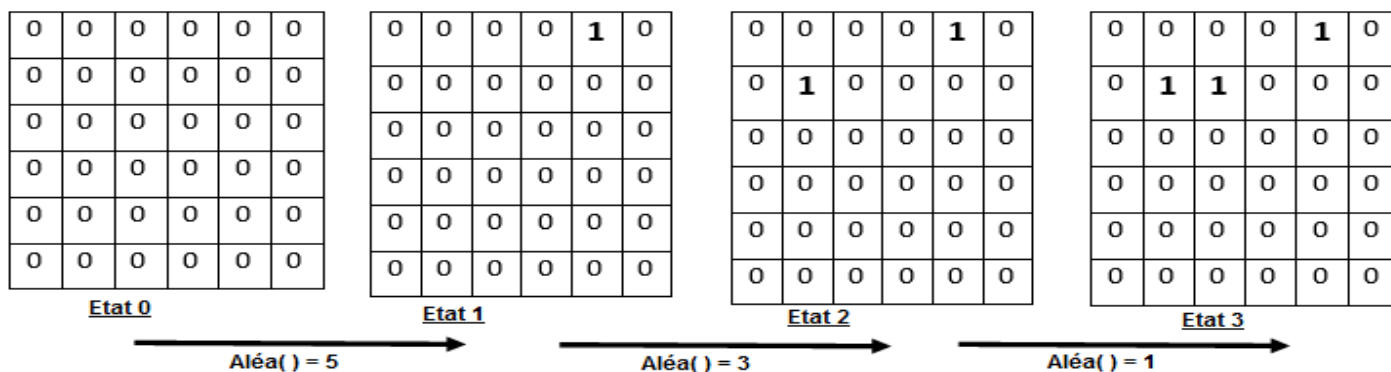
Le déplacement d'une case à une autre est exprimé en nombre de cases à avancer. Ce nombre sera donné de façon aléatoire entre 1 et 6, par la fonction prédéfinie RANDOM.

Le principe d'avancement sur la matrice est le suivant :

- ✚ Si le déplacement de k cases est réalisable sur une même ligne alors le chevalier passe de la case $[i,j]$ vers la case $[i,j+k]$
- ✚ Si la ligne ne suffit pas, le déplacement doit terminer toute la ligne en cours avant de passer à la ligne suivante
- ✚ Dans le cas où le nombre aléatoire k dépasse le nombre de cases restantes dans la matrice, le chevalier n'effectue aucun déplacement et doit attendre son prochain tour pour réessayer de nouveau.

Les deux joueurs jouent d'une **façon alternative** : chacun joue une fois, réalise son déplacement puis il cède le rôle à son adversaire.

Exemple : Evolution de l'état de la matrice d'un joueur pour des tirages aléatoires 5, 3 puis 1.



On se propose d'écrire un programme permettant de simuler ce jeu. Pour cela, on doit suivre les actions suivantes :

- Saisir la dimension n ($6 \leq n \leq 20$) des deux matrices M1 du joueur1 et M2 du joueur2.
- Initialiser les matrices M1 et M2 par des zéros
- Repositionner un chevalier sur M1 ou M2 après un tirage effectué par un joueur. La nouvelle position sera calculée en respectant les contraintes de déplacement décrites ci-dessus.
- Afficher la matrice M1 ou M2 après chaque tirage effectué respectivement par le joueur1 ou le joueur2.
- Afficher le joueur gagnant (joueur1 ou joueur2)
- Enregistrer dans un fichier texte intitulé damier.txt placé sur la racine du lecteur C, l'ensemble des informations relatives au joueur gagnant. Ce fichier comportera :
 - Sur sa première ligne, le nom du joueur gagnant (joueur1 ou joueur2)
 - Les lignes suivantes contiennent la matrice M1 ou M2 du joueur gagnant
 - Sur la dernière ligne, le nombre total des déplacements effectués durant la partie par le joueur gagnant.

Travail demandé :

- 1) Proposer une analyse et un algorithme au problème en le décomposant en modules.
- 2) Analyser chacun des modules envisagés précédemment.



Bon travail

Correction Devoir de Synthèse n°1

Algorithmiques et Programmation 4 SI

Exercice n°1 : (3 points)

Q1 : Exécution à la main :

1 pt

St	Test (St="")	Appel	Résultat
"RADAR"	Non		"RADAR" 1 pt
"RADA"	Non	"R" + FN Inconnue ("RADA")	
"RAD"	Non	"RA" + FN Inconnue ("RAD")	
"RA"	Non	"RAD" + FN Inconnue ("RA")	
"R"	Non	"RADA" + FN Inconnue ("R")	
""	Oui (arrêt)	"RADAR" + ""	

St	Test (St="")	Appel	Résultat
"algo"	Non		"ogla"
"alg"	Non	"o" + FN Inconnue ("alg")	
"al"	Non	"og" + FN Inconnue ("al")	
"a"	Non	"ogl" + FN Inconnue ("a")	
"	Oui (arrêt)	"ogla" + ""	

Q2 : cette fonction permet d'inverser une chaîne de caractères donnée.

1 pt

-0.25 par erreur

Exercice n°2 : (5 points)

Q1 : Algorithme de la fonction Palindrome

- ```

0) DEF FN Palindrome (T : TAB, i, j : entier) : booléen
1) Si i > j Alors Palindrome ← Vrai
 Sinon Si T[i] <> T[j] Alors Palindrome ← Faux
 Sinon Palindrome ← FN Palindrome(T, i+1, j - 1)
 Fin si
 Fin si
2) fin Palindrome

```

0.5

0.5

0.5

0.5

0.5

T.D.O.L :

| Objet | T/N    | Rôle     |
|-------|--------|----------|
| k     | Entier | Compteur |

Q2 : Algorithme de la procédure transfert

- ```

0) DEF Proc Transfert (var F : texte ; T : TAB ; i, j : entier)
1) Associer (F, "C:\fichpal.dat")
2) Recréer (F)
3) Si Palindrome (T,i,j) Alors
   Pour k de i à j faire
     Ecrire_nl ( F , T[k])
   Fin pour
   Fin si
4) Fermer(F)
5) Fin
    
```

0.5

0.25

0.25

0.5

0.25

0.5

0.25

Exercice n°3 : (12 points)

1) Analyse du P.P :

Résultat : F rempli

Traitement : proc remplir (f, M, nb, gagnant)

Proc jouer (M1, M2, M, n, nb, gagnant)

Proc saisir (M1, M2, n)

Associer (f, "C:\damier.txt")

Fin jeu .

1.25

Algorithme du P.P : 0.5

- ```

0) Début jeu
1) saisir(M1,M2,n)
2) jouer (M1,M2, M, n, Nb, gagnant)
3) associer(f, "C:\damier.txt")
4) remplir(f, M, nb, gagnant)
5) fin jeu

```

T.D.N.T :

| Type                                  |
|---------------------------------------|
| <b>Mat</b> = tableau de n x n Entiers |

0.25

T.D.O.G : 0.75

| O       | T / N     | R                                       |
|---------|-----------|-----------------------------------------|
| M1, M2  | Mat       | Matrice joueur1 et joueur 2             |
| M       | Mat       | Matrice du joueur gagnant               |
| N       | Entier    | Taille de la matrice                    |
| Gagnant | Chaîne    | Le gagnant du jeu                       |
| Nb      | Entier    | Nombre total de déplacements            |
| F       | Texte     | Fichier texte                           |
| Saisir  | Procédure | Saisir et initialiser M1 et M2          |
| Jouer   | Procédure | par des zéros                           |
| Remplir | procédure | Appliquer le principe du jeu            |
|         |           | Enregistrer les infos du gagnant dans F |

## 2) Analyse des Modules :

### Analyse de la procédure Saisir:

DEF Proc Saisir (var M1, M2 ; Var n : entier)

Résultat : n, M1, M2

Traitement : [] pour i de 1 à n faire

    Pour j de 1 à n faire

        M1[i,j] ← 0

        M2[i,j] ← 0

    Fin pour

Fin pour

Répéter

    N = donnée

Jusqu'à (n dans [4..20])

Fin Saisir

T.D.O.L :

0.25

| Objet | Type/Nature | Rôle                      |
|-------|-------------|---------------------------|
| i, j  | Entier      | Compteurs lignes/colonnes |

### Analyse de la procédure déplacer:

DEF Proc Déplacer (var M : MAT ; n : entier ; var i, j, nb : entier) ;

Résultat : M, i, j, nb

Traitement : [k ← Aléa(6) + 1]

    Si  $j + k \leq n$  Alors

        J ← j + k

        M[i,j] ← 1

        Nb ← nb + 1

    Sinon Si (j + k > n) ET (i < n) Alors

        I ← i + 1

        J ← k - (n - j)

        M[i,j] ← 1

        Nb ← nb + 1

    Fin Si

Fin Déplacer

0.5

0.75

0.25

### Analyse de la procédure jouer:

DEF Proc Jouer (M1, M2 : MAT ; var M : MAT ; n : entier ; var nb : entier ; var gagnant : chaîne)

Résultat : M1, M2, nb, gagnant

Traitement : Si M1[n,n] = 1 Alors

    gagnant ← "joueur1"

    M ← M1

    Nb ← nb1

    Sinon

    gagnant ← "joueur2"

    M ← M2

    Nb ← nb2

Fin Si

[i1 ← 1, j1 ← 0, nb1 ← 0, i2 ← 1, j2 ← 0, nb2 ← 0]

Répéter

    Proc Déplacer (M1, n, i1, j1, nb1)

    Proc Afficher (M1, n)

    Proc Déplacer (M2, n, i2, j2, nb2)

    Proc Afficher (M2, n)

Jusqu'à (M1[n,n] = 1) ou (M2[n,n] = 1)

Fin jouer

T.D.O.L :

| Objet    | Type/Nature | Rôle                                   |
|----------|-------------|----------------------------------------|
| i1, j1   | Entier      | Compteurs lignes/colonnes M1           |
| i2, j2   | Entier      | Compteurs lignes/colonnes M2           |
| nb1      | Entier      | Nombre de déplacements du joueur1      |
| nb2      | Entier      | Nombre de déplacements du joueur2      |
| Déplacer | Procédure   | Déplacer un chevalier dans une matrice |
| Afficher | Procédure   | Afficher une matrice                   |

0.5

T.D.O.L :

| Objet | T/N    | Rôle                       |
|-------|--------|----------------------------|
| k     | Entier | Nombre de cases à déplacer |

**Analyse de la procédure afficher:**

DEF Proc afficher (M : MAT ; n : entier)

Résultat : afficher M

Traitement : Pour i de 1 à n faire

Pour j de 1 à n faire

Ecrire (M[i,j] ,” “)

Fin pour

Fin pour

Fin Afficher

**0.25****0.5****Analyse de la procédure remplir:**DEF Proc Remplir (var F : texte ; M : MAT ; nb : entier ; gagnant : chaine) **0.25**

Résultat : F rempli

Traitement : fermer(F)

Ecrire\_nl ( F, ‘’ Le nombre total de déplacements est : ‘’, nb)

Pour l de 1 à n faire

Ligne ← ‘’’

Pour c de 1 à n faire

Convch(M[l,c] , ch)

Ligne ← ligne + ‘’ ’’ + ch

Fin pour

Ecrire\_nl ( F, Ligne)

Fin pour

Ecrire\_nl ( F, gagnant )

Ecrire( ‘’le gagnant est : ‘’, gagnant)

Fin Remplir

**0.25****0.25****0.5****0.25****0.25****0.25****0.25****T.D.O.L :**

| Objet | Type/Nature | Rôle                        |
|-------|-------------|-----------------------------|
| i, j  | Entier      | Compteurs lignes/colonnes M |

**0.25****T.D.O.L :**

| Objet | Type/Nature | Rôle                        |
|-------|-------------|-----------------------------|
| l     | Entier      | Compteur des lignes M       |
| c     | Entier      | Compteur des colonnes M     |
| ligne | chaine      | Ligne de matrice en chaine  |
| ch    | chaine      | Valeur d’une case en chaine |