

Exercice 2 (3 points)

Soit l'algorithme suivant de la fonction intitulée **Quoi** :

```
0) DEF FN Quoi (T : Tab ; a, deb, fin : entier) : .....
1) m ← (deb + fin) Div 2
2) Si (deb > fin) Alors Quoi ← faux
   Sinon Si (T[m] = a) Alors Quoi ← vrai
     Sinon Si (T[m] > a) Alors Quoi ← FN Quoi (T, a, deb, m-1)
       Sinon Quoi ← FN Quoi (T, a, m+1, fin)
   Fin Si
3) Fin Quoi
```

NB : T est un tableau d'entiers trié dans l'ordre croissant.

Travail demandé :

- 1- Donner le type de retour de la fonction **Quoi**.
- 2- Donner la valeur retournée par la fonction **Quoi** (T, 6, 1, 5) pour les deux cas suivants :

Cas 1 : T

-2	3	6	10	12
----	---	---	----	----

Cas 2 : T

-2	3	4	10	12
----	---	---	----	----

- 3- Dédurre le rôle de la fonction **Quoi**.

Exercice 3 (4 points)

Le jeu des tours de Hanoï est un jeu de réflexion qui consiste à déplacer N disques de diamètres différents d'une tour de départ (**D**) à une tour d'arrivée (**A**) en passant par une tour intermédiaire (**Inter**), et ceci en un minimum de coups, tout en respectant les règles suivantes :

- On ne peut pas déplacer plus d'un disque à la fois.
- On ne peut placer un disque que sur un autre disque plus grand ou sur un emplacement vide.

Soit l'algorithme récursif suivant de la procédure qui simule le jeu des tours de Hanoï :

```
0) DEF PROC Hanoi (N : entier ; D, A, Inter : caractère)
1) Si (N > 0) Alors
   PROC Hanoi (N-1, D, Inter, A)
   Ecrire ("Déplacer un disque de ", D, " à ", A)
   PROC Hanoi (N-1, Inter, A, D)
   Fin Si
2) Fin Hanoi
```

Travail demandé :

- 1) Pour sauvegarder les différents déplacements des N disques, on se propose d'utiliser un fichier d'enregistrements intitulé "**Deplace.dat**".
 - a) Proposer une déclaration d'un type enregistrement permettant de représenter un déplacement, sachant que chaque déplacement est caractérisé par une tour de départ et une tour d'arrivée.
 - b) Proposer une déclaration d'un type pour le fichier "**Deplace.dat**".

- 2) Réécrire l'algorithme de la procédure **Hanoi** en apportant les modifications nécessaires pour qu'il procède à la sauvegarde des différents déplacements des disques dans le fichier d'enregistrements "**Deplace.dat**".
NB : Le fichier résultat "**Deplace.dat**" est créé et ouvert dans le programme appelant.
- 3) Ecrire un algorithme d'un module permettant d'afficher le contenu du fichier d'enregistrements "**Deplace.dat**".

Problème (11 points)

Parmi les méthodes de cryptage utilisées pour crypter un texte **msg** formé de caractères alphabétiques, on cite la méthode **OU exclusif simple** utilisant une **clef** de cryptage, qui est une chaîne binaire représentée sur **N** bits avec **N** est un multiple de 7.

Cette méthode consiste à réaliser les étapes suivantes :

Etape1 :

Sachant que chaque caractère de **msg** est représenté sur 7 bits, on procède à un ajout du caractère antislash "\" à la fin de **msg**, autant de fois que nécessaire, jusqu'à ce que la longueur de sa représentation binaire devienne un multiple de **N**.

Etape2 :

On découpe **msg** en blocs de caractères pour que chacun soit représenté sur un nombre de bits égal au nombre de bits de la clef, c'est-à-dire que chaque bloc contient **N Div 7** caractères.

Etape3 :

On détermine la représentation binaire de chaque bloc par la concaténation de l'équivalent binaire, sur 7 bits, du code Ascii de chaque caractère du bloc.

Etape4 :

A chaque bloc, on applique un **Ou exclusif (OUex)** avec la clef, bit par bit.
On rappelle que $0 \text{ OUex } 0 = 0$, $1 \text{ OUex } 0 = 1$ et $1 \text{ OUex } 1 = 0$

Etape5 :

Le texte crypté est obtenu en concaténant respectivement les résultats trouvés dans l'étape précédente.

Exemple :

Pour **clef** = "10101100101101" et **msg** = "Savon"

Etape1 :

Puisque le texte à crypter est formé de 5 caractères et chaque caractère est représenté sur 7 bits, la longueur de la représentation binaire du texte est 35 (5 caractères * 7 bits).

Comme la longueur de la clef est égale à 14 et comme 35 n'est pas un multiple de 14, il est nécessaire d'ajouter à droite du texte le caractère antislash "\" une seule fois afin que la nouvelle longueur ($6*7=42$) devienne un multiple de 14.

Etape2 :

Etant donné le texte modifié (**msg** = "Savon\"), on procède à un découpage en blocs de 14 bits c'est à dire en blocs de deux caractères (14 Div 7).

D'où, Bloc1 = "Sa", Bloc2 = "vo" et Bloc3 = "n\"

