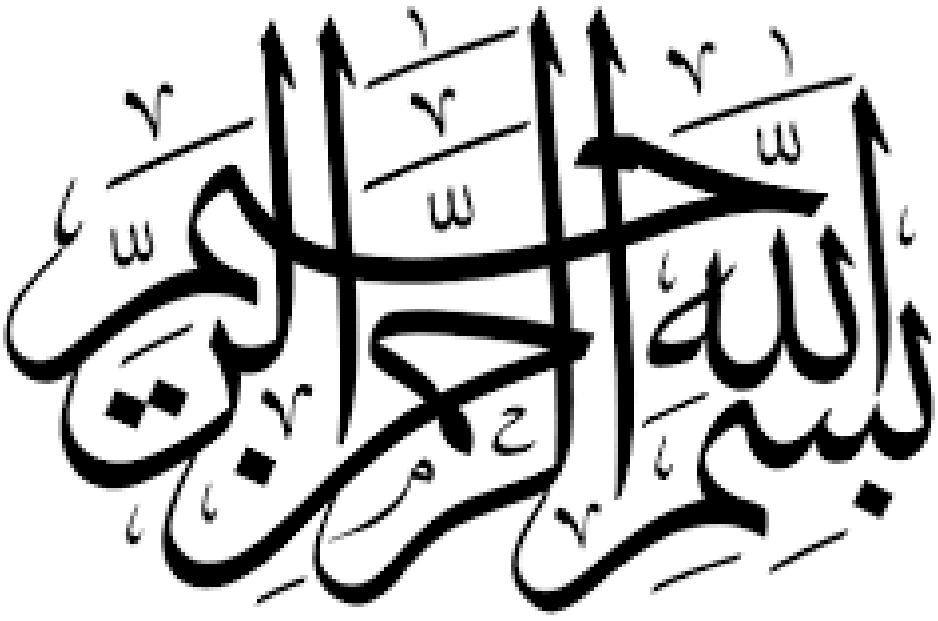
**Brahim BESSAA**

# Algorithmique

## *Exercices avec Solutions*

***1ére Année MI***

**Septembre 2017**


بسم الله الرحمن الرحيم

# *Préface*

Après quelques années d'enseignement du module « Algorithmique » de la première année licence (MI) et vue les difficultés trouvées par les étudiants dans ce module, j'ai essayé de mettre à leur disposition un support d'entrainement afin de les aider à maitriser ce module.

Cet ouvrage regroupe des exercices des séries des travaux dirigés et examens (avec corrigés) du module Algorithmique de la première année MI (USTHB). Dans cet ouvrage je donne des solutions détaillées aux exercices proposés, mais il ne doit en aucun cas remplacer les séances de TD, où les étudiants peuvent discuter les solutions et voir d'autres propositions de solutions. En fait, le chargé du TD peut toujours donner plus de détails et d'explications.

Une exploitation positive de cet ouvrage consiste donc à pousser les étudiants à préparer leurs séries d'exercices, comparer leurs solutions avec celles proposées et prévoir des questions à poser lors des séances de TD.

Enfin, l'ouvrage est une première version d'un effort personnel. J'attends des chers étudiants et collègues leurs remarques et suggestions afin de l'améliorer dans les prochaines versions.

Septembre 2017.

Dr. Brahim BESSAA

bbessaa@yahoo.fr

# *Sommaire*

**EXERCICE 1**

Ecrire un algorithme qui demande un nombre à l'utilisateur, puis calcule et affiche le carré de ce nombre.

```
Algorithme Carre ;
Var    X,X2 :reel ;
Début
       Ecrire('Donner un reel') ;
       Lire(X) ;
       X2←X*X ;
       Ecrire('Le carré de ', X,' est: ',X2) ;
Fin.
```

**EXERCICE 2**

Un magasin de reprographie facture 2 DA les dix premières photocopies, 1.50 DA les vingt suivantes et 1 DA au-delà. Ecrivez un algorithme qui demande à l'utilisateur le nombre de photocopies effectuées puis affiche le montant correspondant.

```
Algorithme Facture ;
Const  P1=2 ; P2=1.5 ; P3=1 ;
Var    Mont :reel ;
       Nbc :entier ;
Début
       Ecrire('Donner le nombre de photocopies') ;
       Lire(Nbc) ;
       Si Nbc≤10
       Alors  Mont←P1*Nbc
       Sinon  Si Nbc≤30
              Alors  Mont←P1*10+P2*(Nbc-10)
              Sinon  Mont←P1*10+P2*20+P3*(Nbc-30)
              Fsi
       Fsi ;
       Ecrire('Le montant à payer est: ',Mont) ;
Fin.
```

**EXERCICE 3**

Ecrire un algorithme permettant d'afficher la saison en introduisant le numéro du mois.

```
Algorithme Saison;
Var    M :entier ;
Début
       Ecrire('Donner un numéro de mois 1--12') ;
       Répéter  Lire(M) ; Jusqu'à M>0 et M<13 ;
```

```
        Cas M Vaut
                3,4,5 : Ecrire('La saison est : PRINTEMPS') ;
                6,7,8 : Ecrire('La saison est : ETE') ;
                9,10,11 : Ecrire('La saison est : AUTOMNE') ;
                12,1,2 : Ecrire('La saison est : HIVER') ;
        FinCas ;
Fin.
```

**EXERCICE 4**

Ecrire un algorithme pour résoudre chacun des problèmes suivants :

1- Calcul de la somme des N premiers nombres entiers.
2- Recherche du minimum et du maximum dans un ensemble de N nombres.
3- Calcul du quotient et reste de la division de deux entiers A et B sans utiliser l'opération de division.
4- Le calcul du produit de deux entiers en utilisant uniquement l'opération d'addition '+'.
5- Détermination si A est divisible par B. Avec A et B des entiers positifs.
6- Déterminer tous les diviseurs d'un entier X donné.
7- Déterminer si un nombre entier X est premier ou non.
8- Calcule la somme des chiffres qui composent un entier naturel N.

1-

```
Algorithme Somme ;
Var     I,N,S :entier ;
Début
        Ecrire('Donner un entier N') ; Lire(N) ;
        S ←0 ;
        Pour  I ←1 à N-1
        Faire
                S ←S+I;
        Fait ;
        Ecrire('La somme des', N,' premiers nombres est: ',S) ;
Fin.
```

2-

```
Algorithme MaxMin;
Var     I,N,Max,Min,X :entier ;
Début
        Ecrire('Donner un entier N>0') ;
        Répéter  Lire(N) ; Jusqu'à N>0 ;
        /* Lire le premier élément, puis initialiser le Min et le Max à cette valeur
        Lire(X) ; Max←X ; Min←X ;
        Pour  I ←2 à N
        Faire
                /* lire la suite des éléments et mettre à jour le Min et le Max
                Lire(X) ;
```

```
            Si Max<X      Alors Max←X
                          Sinon Si Min>X Alors Min←X Fsi
            Fsi ;
      Fait ;
      Ecrire('Le Minimun des valeurs est: ',Min,' le Maximum est : ',Max) ;
Fin.
```

3-

```
Algorithme QuotReste ;
Var   A,B,Q,R :entier ;
Début
      Ecrire('Donner deux entiers A et B') ;
      Lire(A,B) ;
      Q ←0 ; R ←A ;
      Tantque R>B
      Faire
            Q ←Q+1;
            R ←R-B;
      Fait ;
      Ecrire('Le Quotient de A/B est : ',Q, ' Le reste de A/Best : ',R) ;
Fin.
```

4-

```
Algorithme Produit ;
Var   A,B,P,I :entier ;
Début
      Ecrire('Donner deux entiers A et B') ;
      Lire(A,B) ;
      Si A=0 ou B=0
      Alors P←0
      Sinon  P←0 ; /*initialiser le produit à 0
            Pour  I ←1 à B
            Faire
                  P←P+A ;
            Fait
      Fsi ;
      Ecrire('Le produit A*B est : ',P) ;
Fin.
```

On peut optimiser la solution en choisissant la boucle ayant le moins d'itérations :

```
Algorithme Produit ;
Var   A,B,P,I :entier ;
Début
      Ecrire('Donner deux entiers A et B') ;
```

```
        Lire(A,B) ;
        Si A=0 ou B=0
        Alors P←0
        Sinon  Si A>B
                Alors  P←A ; /*On peut initialiser le produit à A et commencer la boucle à 2
                        Pour  I ←2 à B
                        Faire
                                P←P+A ;
                        Fait
                Sinon  P←B ;
                        Pour  I ←2 à A
                        Faire
                                P←P+B ;
                        Fait
        Fsi ;
        Ecrire('Le produit A*B est : ',P) ;
Fin.
```

5-

```
Algorithme AdivB;
Var     A,B,R :entier ;
Début
        Ecrire('Donner deux entiers positifs A,B') ;
        Répéter  Lire(A,B) ; Jusqu'à A>0 et B>0 ;
        R←A ;
        Tantque R≥0 Faire   R ←R-B; Fait ;
        Si R=0 Alors Ecrire(A,' est divisible par ',B)
                Sinon Ecrire(A,' est n''est pas divisible par ',B)
        Fsi ;
Fin.
```

6-

```
Algorithme Diviseurs ;
Var     X,M,I :entier ;
Début
        Ecrire('Donner un entier X') ;
        Lire(X) ;
        Ecrire('Les diviseurs de ',X,' sont :') ;
        /*On boucle de 1 à la moitié de X, car après la moitié il n'y a plus de diviseur sauf X
        /*On peut utiliser la fonction division entière DIV et la fonction reste de cette division MOD
        M←X DIV 2 ;
        Pour  I ←1 à M
        Faire
                Si X MOD I=0 Alors Ecrire(I) Fsi;
        Fait ;
```

```
        Ecrire(X) ;
Fin.
```

7-

```
Algorithme Premier;
Var     X,M,I :entier ;
        Pr :booléen ;
Début
        Ecrire('Donner un entier X') ;
        Lire(X) ;
        /*X est premier s'il a deux diviseurs distincts 1 et lui-même, attention 1 n'est pas premier.
        Pr←Vrai ;
        Si X=1
        Alors Pr←Faux
        Sinon  M←X DIV 2 ;
                I ←2 ;
                Tantque  I ≤ M et Pr
                Faire  /*si on trouve un diviseur on arrête la boucle
                        Si X MOD I=0 Alors Pr←Faux Fsi;
                        I ←I+1 ;
                Fait
        Fsi ;
        Si Pr Alors Ecrire(X,' est premier') Sinon Ecrire(X,' n''est pas premier') Fsi;
Fin.
```

8-

```
Algorithme SommeChiff;
Var     N,S,R :entier ;
Début
        Ecrire('Donner un entier naturel N') ;
        Répéter  Lire(N) ; Jusqu'à N≥0 ;
        S←0 ; R←0 ;
        Tantque  R>0
        Faire  S←S+R MOD 10;
                R← R DIV 10;
        Fait ;
        Ecrire('La somme des chiffres qui composent ',N,' est :',S) ;
Fin.
```

**EXERCICE 5**

Ecrire un algorithme qui permet à l'utilisateur de saisir une suite caractère se terminant par '*****', et qui affiche à la fin le nombre d'apparition de la lettre '**A**'.

**Solution 1** : en utilisant une boucle **Répéter**

```
Algorithme Appatition ;
```

```
Var     ch :caractère ;
        NbA :entier ;
Début
        NbA ←0 ;
        Répéter
                Lire(ch) ;
                Si ch='A' Alors NbA ←NbA+1 Fsi ;
        Jusqu'à ch='*' ;
        Ecrire('Nombre apparition de A est :',NbA) ;
Fin.
```

**Solution 2** : en utilisant une boucle **Tantque +Initialisation**

```
Algorithme Appatition ;
Var     ch :caractère ;
        NbA :entier ;
Début
        NbA ←0 ;
        Ch ←'X' ; /* Initialiser Ch à un caractère autre que '*'
        Tanque ch<>'*'
        Faire
                Lire(ch) ; /* la lecture se fait avant le traitement
                Si ch='A' Alors NbA ←NbA+1 Fsi ;
        Fait ;
        Ecrire('Nombre apparition de A est :',NbA) ;
Fin.
```

**Solution 3** : en utilisant une boucle **Tantque + Lecture avant la boucle**

```
Algorithme Appatition ;
Var     ch :caractère ;
        NbA :entier ;
Début
        NbA ←0 ;
        Lire(ch) ; /* lecture la première valeur de  ch avant la boucle
        Tanque ch<>'*'
        Faire
                Si ch='A' Alors NbA ←NbA+1 Fsi ;
                Lire(ch) ; /* La lecture suivante se fait après le traitement
        Fait ;
        Ecrire('Nombre apparition de A est :',NbA) ;
Fin.
```

**EXERCICE 6**

Ecrire un algorithme permettant de calculer la valeur de l'expression E,
telle que E=(1+2)x(1+2+3)x(1+2+3+4)x…x(1+2+3+…+(N-2)+(N-1)+N), et (N≥2).

```
Algorithme SommeE ;
Var  I,J,N,E,S :entier ;
Début
        Lire(N) ;
        E ←1 ;
        S ←1;
        Pour  I ←2 à N
        Faire
                S ←S+I;
                E ←E*S ;
        Fait ;
        Ecire('E=',E) ;
Fin.
```

**EXERCICE 7**

Ecrire un algorithme permettant de calculer la valeur de l'expression E,

$$\boldsymbol{E = 1 + \frac{1}{1+2} + \frac{1}{1+2+3} + \cdots + \frac{1}{1+2+3+\cdots+N}}$$ , avec (N≥2).

```
Algorithme SommeE ;
Var  I,J,N,S :entier ;
        E:reel;
Début
        Répéter Lire(N) ; Jusqu'à N>=2 ;
        E ←1 ;S ←1;
        Pour  I ←2 à N  Faire
                                S ←S+I;
                                E ←E+1/S ;
                        Fait ;
        Ecire('E=',E) ;
Fin.
```

**EXERCICE 8**

Ecrire un algorithme permettant de calculer la valeur du Nème terme (N<100) de la suite $U_N$ définie par :

$$U_0 = 2,\ U_1 = 3,\ U_{N+2} = \frac{2}{3}U_{N+1} - \frac{1}{4}U_N$$

```
Algorithme Suite ;
Var     I,N :entier ;
        X,Y,Un :reel ;
```

```
Debut
    Ecrire ('Donner  0<N<100') ;
    Repeter Lire(N) Jusqu'à (N>0) et (N<100) ;
    X← 2 ;
    Y←3 ;
    /* Attention le 1 er terme (N=1) correspond à U0, la récurrence commence à partir de 3
    Pour I←3 à N
    Faire
        Un ← (2/3)*Y-(1/3)*X ; /* il faut linéariser l'expression
        X ← Y ;
        Y ← Un ;  /*attention à l'ordre entre les deux dernières affectations
    Fait ;
    Cas N Vaut
    1 : Un ← X ;
    2 : Un ← Y ;
    FinCas ;
    Ecrire('Le ',N,' ème terme est : ',Un) ;
Fin.
```

**EXERCICE 9**

Ecrire un algorithme qui détermine et affiche la $N^{ème}$ valeur de la suite $(U_N)$ sachant que

$$U_0 = 0 \quad U_1 = 1 \; ; \; U_2 = 2 \; ; \quad \mathbf{U_N = U_{N-1} + U_{N-3}} \quad \text{pour } N > 2.$$

```
Algorithme suite;
Var X,Y,Z,Un,I,N:entier;
Debut
    Ecrire('Donner un entier') ;
    Repeter Lire(N) ; Jusqu'à N≥0 ;
    X←0 ;
    Y←1 ;
    Z←2 ;
    Pour I←3 à N Faire
                    Un←Z+X ;
                    X←Y ;  Y←Z ; Z←Un ;
                Fait ;
    Cas N Vaut
    0 : Un←X ;
    1 : Un←Y ;
    2 : Un←Z ;
    Fincas ;
    Ecrire('Le terme Un est :',Un) ;
Fin.
```

**EXERCICE 10**

Ecrire un algorithme permettant de convertir un entier N écrit sous forme binaire en sa valeur décimale.

**Exemple** : N =10111010 après conversion on obtient valeur décimale = 186

```
Algorithme conversion ;
Var   VB,B,D,P2 :entier ;
Debut
      Ecrire('Donner un entier en binaire') ;
      Repeter Lire(VB) Jusqu'à VB>=0 ;
      B ← VB ;  /* sauvegarde de VB pour affichage
      P2 ← 1 ;   /* P2 contient la puissance de 2, initialement 2^0=1
      D ← 0 ;
      Repeter
          D ← D + (B MOD 10)*P2 ; /*récupérer le coefficient= le chiffre le plus à droite du nombre
          P ← 2 * P2 ;  /* calcul de la puissance suivante de 2
          B ← B DIV 10 ; /* pour passer au coefficient suivant
      Jusqu'à B=0 ;
      Ecrire('La valeur décimale de ', VB, ' est : ',D) ;
Fin.
```

**EXERCICE 11**

Ecrire un algorithme qui calcule la somme d'ordre N de $\boldsymbol{Sn}$ définie comme suit en utilisant seulement les opérateurs de base (**sans l'utilisation de l'opérateur de puissance**).

$$\boldsymbol{Sn = \sum_{i=0}^{N} \frac{(-1)^{i+1}}{x^i}}$$

```
Algorithme SommeSuite;
Var   I,N,K:entier;
      Px,X,Sn :reel ;
Debut
    Ecrire('Donner un entier') ;
    Repeter Lire(N) Jusqu'à N>=0 ;
    Ecrire('Donner un réel') ;
    Lire(X) ;
    K←-1 ;   /* K représente (-1)^i
    Sn←-1 ;  /*valeur initiale de Sn pour i=0
    Px←1 ;   /*puissance de X pour i=0
    Pour I←1 à N
    Faire
          K← - K;
          Px←Px*X ;
```

```
            Sn←Sn+K/Px ;
    Fait ;
    Ecrire('La somme Sn= ', Sn) ;
Fin.
```

**EXERCICE 1**

Ecrire les actions paramétrées (procédure ou fonction) permettant de résoudre les problèmes suivants :

1- Calcul de la somme de deux nombres entiers.
2- Calcul de la factorielle de N (N !).
3- Vérifier si un nombre entier A divise un nombre entier B.
4- Calcul du quotient et du reste de la division entière de deux nombres entiers A et B.
5- Vérifier si un caractère donné est une voyelle (voyelles : 'a', 'e', 'i', 'o', 'u', 'y').
6- Permet de permuter (d'échanger) le contenu de deux variables réelles.
7- Etant donné un entier A, calcule sa valeur absolue.

```
1- Fonction Somme(x,y :entier) :entier ;
   Debut
        Somme ← x+y ;
   Fin ;
2- Fonction Fact(x:entier) :entier ;
   Var   I,F :entier ;
   Debut
        F←1 ; /* on peut utiliser directement le nom de la fonction au lieu de F
        Pour  I←1 à x
        Faire F ← F*I ; Fait ;
        Fact ← F ;
   Fin ;
3- Fonction Divise(A,B :entier) :booleen ;
   Debut
        Divise ← Faux ;
        Si B mod A = 0 Alors Divise ← Vrai Fsi ;
   Fin ;
4- Procedure QuotRest(E/ A,B :entier ; S/ Q,R :entier) ;
   Debut
        Q ← 0 ;  R ← A ;
        Tantque R>= B
        Faire
        R ← R mod B ;
        Q ← Q+1 ;
        Fait ;
   Fin ;
5- Fonction Voyelle(C :caractère) :booleen ;
   Debut
        Voyelle ← Faux ;
        Cas C Vaut
         'a', 'e', 'i', 'o', 'u', 'y': Voyelle ← Vrai ;
        Fincas ;
   Fin ;
6- Procedure Permute(E/S/ A,B :entier ) ;
   Var C:entire:
   Debut
```

```
        C ← A ;  A ← B ;  B ← C ;
    Fin ;
7-  Fonction Vabs(A :entier) :entier ;
    Debut
        Vabs ←  A ;
        Si A<0 Alors Vabs ←  A Fsi;
    Fin ;
```

**EXERCICE 2**

1- Ecrire une AP **Carre v**érifiant si un nombre entier naturel est un carré parfait, en utilisant seulement les opérateurs de base, et renvoie sa racine dans le cas favorable. (Indication : X est un carré parfait s'il existe un entier i tel que X = i * i.)

2- Ecrire un algorithme qui, parmi N entiers naturels, calcul la somme et le produit des racines carrées des entiers carrés parfaits. Ensuite il vérifie si la somme et le produit sont des carrés parfaits.

```
1-  Procedure  Carre(E/ A:entier ; S/ CP ::booleen; S/ RC:entier ) ;
    Var     I:entier ;
    Debut
        CP← Faux ; I ← 0 ;
        Tantque  (I<= A div 2)et(Non CP)
        Faire  Si A=I*I Alors CP← Vrai ; RC ← I Fsi ;
        Fait ;
    Fin ;
2-  Algorithme Calcul ;
    Var I,N,S,P,X,Rac :entier ;  CParfiat:boolen;
    Procedure  Carre(E/ A:entier ; S/ CP ::booleen; S/ RC:entier ) ;
        /* on reprend la déclaration de la procédure
        -   - - - -
    Debut
    Ecrire('Donner le nombre d''élements N') ;
    Repeter Lire(N) Jusqu'à N>0 ;
        S← 0 ; P ← 1 ;
        Pour I ← 1 à N
        Faire   Lire(X) ;
        Carre(X,Cparfait,Rac) ;
            Si CParfait  Alotrs
                        S← S+Rac ;
                        P← P*Rac
            Fsi ;
        Fait ;
        Carre(S,Cparfait,Rac) ;
        Si CParfait Alors Ecrire('La somme S=',S,' est un carré parfait') Fsi ;
        Carre(P,Cparfait,Rac) ;
        Si CParfait  Alors Ecrire('Le produit=',P,' est un carré parfait') Fsi ;
    Fin.
```

**EXERCICE 3**

1- Ecrire une fonction qui retourne Vrai si le caractère passé en paramètre est égal à ′o′ ou ′O′ (qui veut dire Oui), et Faux sinon.

2- Ecrire une action paramétrée qui permet d'afficher la table de multiplication de 1 à 9 d'un nombre entier positif. Puis, en utilisant les actions paramétrées précédentes, écrire un algorithme permettant d'afficher à l'utilisateur la table de multiplication d'un entier aussi longtemps qu'il le désire (jusqu'à ce que la réponse soit fausse).

```
1- Fonction Reponse(C:caractère) :booleen ;
   Debut
        Reponse ← Faux ;
        Si C='o' ou C='O ' Alors Reponse ← Vrai Fsi ;
   Fin ;
2- Procedure AfficheTable(A :entier) ;
   Var  I :entier ;
   Debut
        Pour I ← à 9
        Faire  Ecrire(A,'x',I,'=',A*I) ; Fait;
   Fin ;
3- Algorithme TableM ;
   Var A :entier ; Rep :caractère ;
       Fonction Reponse(C:caractère) :booleen ;
        . . . . . .
       Procedure AfficheTable(A :entier) ;
        . . . . . .

   Debut
        Repeter
             Ecrire('Donner un entier') ;
             Lire(A) ;
             AfficheTable(A) ;
             Ecrire('Voulez vous continuez O/N') ;
             Lire(Rep) ;
        Jusqu'à Non Reponse(rep) ;
   Fin.
```

**EXERCICE 4**

Ecrire un algorithme affichant tous les nombres inférieurs à 500 égaux à la somme des cubes de leurs chiffres. On utilisera une fonction UNITE, et une fonction CUBE.

Exemple : $153 = 1^3 + 5^3 + 3^3 = 1 + 125 + 27$

```
Algorithme SommeCube ;
Var A,B,S :entier ;
    Fonction Unite(X :entier) :entier ;
    Debut
         Unite ← X mod 10 ;
```

```
    Fin ;
    Fonction Cube(X :entier) :entier ;
    Debut
            Cube ← X*X*X ;
    Fin ;
Debut
      Pour A← 0 à 500
      Faire
            S← 0; B ← A;
            Repeter S← S + Cube(Unite(B)); B ← B div 10 Jusqu'à B=0;
            Si S=A Alors Ecrire(A,' est égale à la somme des cubes de ses chiffres') Fsi ;
      Fait ;
Fin .
```

**EXERCICE 5**

Ecrire un algorithme affichant tous les nombres parfaits inférieurs à 10000. Sachant qu'un nombre entier positif (N) est parfait s'il est égal à la somme de ses diviseurs (<N). On écrira une fonction booléenne, appelée PARFAIT, pour vérifier si le nombre est parfait ou non parfait.
Exemples : 6 — qui est égal à 1 + 2 + 3 —
et 28 — qui est égal à 1 + 2 + 4 + 7 + 14 — sont des nombres parfaits.

```
Algorithme NombreParfait ;
Var A :entier ;
    Fonction Parfait(X :entier) :booleen ;
    Var I,S :entier ;
    Debut
       S ← 0 ;
       Pour I ← 1 à X div 2
       Faire  Si X mod I = 0 Alors S ← S+I Fsi ;
       Fait;
       Si S=X Alors Parfait ← Vrai Sinon Parfait ← Faux Fsi;
    Fin ;
Debut
      Pour A← 1 à 10000
      Faire  Si Parfait(A) Alors Ecrire(A,' est parfait') Fsi ;
      Fait ;
Fin .
```

**EXERCICE 6**

Ecrire une fonction ( MINUS ou LWCASE ) qui permet de convertir un caractère majuscule en son correspondant minuscule.

Pour cet exercice, on n'a pas de fonction prédéfinie en algorithmique traitant les caractères (suivant, précédent, code du caractère, caractère du code,..), donc ou bien qu'on utilise un **Cas Vaut** avec les 26 cas possibles, ou bien on utilise une boucle comme suit.

```
Fonction Minus(C :caractère) :caractère ;
Var I,J :entier ; X :caractère ;
Debut
        /*recherche de la position de C dans l'alphabet
        I ← 0 ;
        Pour X ← 'A' à C Faire I ← I+1 ; Fait ;
        J ← 1 ;
        Pour X ← 'a' à 'z' Faire Si I=J Alors Minus ← X Fsi ; J ← J+1 ; Fait ;
Fin ;
```

**EXERCICE 7**

1- Ecrire une AP **Premier** qui détermine si un entier positif est premier.
2- Ecrire une AP **SPremier** qui détermine si un entier positif est semi-premier.*(un nombre semi-premier est un produit de deux nombres premiers non nécessairement distincts.).*
3- En utilisant l'AP définie précédemment, écrire un algorithme qui détermine les nombres semi-premiers parmi les entiers de la forme (qui s'écrivent de la manière suivante) abcabc où a,b,c sont des chiffres entre 0 et 9 avec a > 0.
Ex : 136136, 524524, 908908, ...

Dans cet exercice, d'après la deuxième question, il est préférable d'écrire une procédure qui revoie le nombre de diviseurs de l'entier en entrée et un booléen pour vérifier s'il est premier ou non.

**1-**
```
Fonction Premier(E/ X : entier ):booleen;
Var I,M:entier;
Debut
        Si X=1 Alors Premier ← Faux
                Sinon  M ← X DIV 2 ; I ← 2 ; Premier ← Vrai ;
                       Tantque I≤M et Premier
                       Faire
                              Si X mod I=0 Alors Premier ← Faux Fsi ;
                              I←I+1 ;
                       Fait ;
        Fsi ;
Fin ;
```

**2-**
```
Fonction SemiPremier(E/ X : entier ):booleen;
Var I,M:entier;
Debut
        Si Premier(X)
        Alors  SemiPremier ← Faux
        Sinon  M ← X DIV 2 ; I ← 2 ;
               SemiPremier ← Vrai ;
               Tantque I≤M et SemiPremier
               Faire
                      Si X MOD I=0
                      Alors  Si (Non Premier(I))ou(Non Premier(X DIV I))
                             Alors SemiPremier ← Faux
                             Fsi
```

```
                    Fsi ;
                    I←I+1 ;
                Fait ;
        Fsi ;
Fin ;
```

**3-** Pour cette question, il s'agit de trouver une formule pour générer les nombre de la forme **abcabc** afin d'optimiser l'algorithme (minimiser le nombre d'itérations).

On a :

$$abcabc = ax10^5 + bx10^4 + cx10^3 + ax10^2 + bx10^1 + cx10^0$$
$$= (10^3 + 1)(ax10^2 + bx10^1 + cx10^0) = 1001(ax10^2 + bx10^1 + cx10^0)$$

Donc pour générer ces nombre il suffit de générer une partie (abc), ensuite la multiplier par 1001.

```
Algorithme Nombre ;
Var a,b,c,X,NBDiv :entier ; Pr :booleen ;
    Fonction Premier(X : entier ): booleen;
        --- --- ---
    Fonction SemiPremier(X : entier ): booleen;
        --- --- ---
Debut
  /* trios boucles imbriquées Pour générer un nombre de la forme abcabc
  Pour a ← 1 à 9
  Faire  Pour b ← 0 à 9
          Faire  Pour c ← 0 à 9
                 Faire
                   X ← 1001*(100*a+10*b+c) ;
                   Si SemiPremier(X) Alors Ecire(X,' est semi premier')  Fsi ;
                 Fait ;
          Fait ;
  Fait ;
Fin.
```

## EXERCICE 8

Ecrire une fonction **BIN** permettant de convertir un entier positif du décimal au binaire.

```
Fonction BIN(X :entier) :entier ;
Var R,P :entier ;
Debut
        P ← 1 ; BIN ← 0 ;
        Repeter
                R ← X mod 2 ;
                BIN ← BIN + R*P ;
                P ← P*10 ;
                X ← X div 2 ;
        Jusqu'à X=0 ;
```

**Fin**.

**EXERCICE 9**

Ecrire une fonction **RACINE2** qui calcule la racine carré d'un nombre positif en utilisant la formule suivante : $X_{i+1} = \frac{1}{2}(X_i + \frac{a}{X_i})$

Où $\sqrt{a} = X_{i+1}$ avec une précision ER = $| X_{i+1} - X_i |$ (ex. ER=$10^{-3}$)

```
Fonction Racine2(A :reel) :reel ;
Const  ER=0.001 ;
Var  X,Y,D :reel ;
Debut
        X ← A ;
        Repeter
                Y ← 0.5*(X+A/X) ;
                D ← Y-X ;
                Si D<0 Alors D ← -D Fsi ;
                X ← Y ;
        Jusqu'à D<ER ;
        Racine2 ← Y ;
Fin ;
```

**EXERCICE 10**

Dérouler l'algorithme suivant en donnant les différentes valeurs des résultats attendus dans l'ordre d'exécution de l'algorithme.

```
Algorithme Appel ;
Var    R , V : entier ;
        Fonction CALCUL ( X : ENTIER ) : entier ;
        Var    R : entier ;
        Debut
                R ← X + V ;
                V← R – 2 ;
                CALCUL ← R + 2 * V ;
                Ecrire ( R , V ) ;
        Fin ;
Debut
        V ← 5 ;
        R ← CALCUL(V) ;   Ecrire ( R , V ) ;
        R ← CALCUL ( V ) ; Ecrire ( R , V ) ;
        R ← 10 ;
        V ← CALCUL ( R ) ;  Ecrire ( R , V ) ;
Fin.
```

| Action | Global | | Fonction | | | Affichage |
|---|---|---|---|---|---|---|
| | R | V | CALCUL | X | R (local) | |
| V ←5 | | 5 | | | | |
| CALCUL(V) | | | | 5 | | |
| R ← X+V | | | | | 10 | |
| V ← R-2 | | 8 | | | | |
| CALCUL ← R+2*V | | | 26 | | | |
| ECRIRE(R,V) | | | | | | 10 8 |
| R← CALCUL | 26 | | | | | |
| ECRIRE(R,V) | | | | | | 26 8 |
| CALCUL(V) | | | | 8 | | |
| R ← X+V | | | | | 16 | |
| V ← R-2 | | 14 | | | | |
| CALCUL ← R+2*V | | | 44 | | | |
| ECRIRE(R,V) | | | | | | 16 14 |
| R← CALCUL | 44 | | | | | |
| ECRIRE(R,V) | | | | | | 44 14 |
| R ← 10 | 10 | | | | | |
| CALCUL(R) | | | | 10 | | |
| R ← X+V | | | | | 24 | |
| V ← R-2 | | 22 | | | | |
| CALCUL ← R+2*V | | | 68 | | | |
| ECRIRE(R,V) | | | | | | 24 22 |
| V ← CALCUL | | 68 | | | | |
| ECRIRE(R,V) | | | | | | 10 68 |

**EXERCICE 1**

Soit un vecteur T (tableau à une dimension) contenant N nombres entiers (N≤100). Ecrire les algorithmes pour :

1- Détermine le minimum, le maximum et la moyenne des éléments d'un tableau T
2- Calcule le produit de tous les éléments de T ainsi que le nombre de valeurs strictement positives.
3- Calcule la somme et le produit scalaire de deux vecteurs (T1 et T2).
4- Détermine les positions de l'apparition d'une valeur dans un vecteur T.
5- Inverse le contenu d'un vecteur T.
6- Supprime toutes les valeurs nulles d'un vecteur T.
7- Met les valeurs négatives au début et les valeurs positives à la fin en utilisant un seul tableau.

1-
```
Algorithme MinMax ;
Var     T :Tableau[1..100] de entier ;
        I,N,Max,Min,S :entier ;  Moy :reel ;
Debut
        /*lecture de la taille exacte
        Ecrire('Donner la taille du tableau N≤100') ;
        Repeter Lire(N) Jusqu'à N>0 et N≤100 ;
        /*Lecture des éléments de T
        Pour I←1 à N Faire Lire(T[I]) ; Fait ;
        /*Initialisation
        Min←T[1] ; Max←T[1] ; S←0 ;
        Pour I←1 à N Faire
                Si Max<T[I] Alors Max←T[I] Fsi ;
                Si Min>T[I] Alors Min←T[I] Fsi ;
                S←S+ T[I] ;
        Fait ;
        Moy←S/N ;
        Ecrire('Maximum=',Max,' Minimum=',Min,' Moyenne=',Moy) ;
Fin.
```

2-
```
Algorithme Prod ;
Var     T :Tableau[1..100] de entier ;
        I,N,P,Nbp :entier ;
Debut
        /*lecture de la taille exacte
        Ecrire('Donner la taille du tableau N≤100') ;
        Repeter Lire(N) Jusqu'à N>0 et N≤100 ;
        /*Initialisation
        P←1 ; Nbp←0 ;
        /*Lecture des éléments de T et traitement en même temps
        Pour I←1 à N Faire
                Lire(T[I]) ;
                Si T[I]>0  Alors Nbp←Nbp+1 Fsi ;
                P←P* T[I] ;
        Fait ;
        Ecrire('Produit=',P,' Nb val positives=',Nbp) ;
Fin.
```

3-
```
Algorithme Prod ;
Var     T1,T2,T3 :Tableau[1..100] de entier ;
        I,N,PS :entier ;
```

```
Debut
        /*lecture de la taille exacte
        Ecrire('Donner la taille du tableau N≤100') ;
        Repeter Lire(N) Jusqu'à N>0 et N≤100 ;
        /*Lecture des éléments de T1 ensuite T2 ne pas lire dans la même boucle
        Pour I←1 à N Faire Lire(T1[I]) ; Fait ;
        Pour I←1 à N Faire Lire(T2[I]) ; Fait ;
        PS←0 ;  /*initialiser Produit scalaire à 0
        /*La somme de T1 et T2 dans T3
        Pour I←1 à N Faire
                T3[I]←T1[I]+ T2[I];
                PS←PS+ T1[I]* T2[I];
        Fait ;
        Ecrire('Produit Scalaire=',PS) ;
        Ecrire('Somme des vecteurs') ;
        Pour I←1 à N Faire Ecrire (T3[I]) ; Fait ;
Fin.
```

4- 
```
Algorithme Position ;
Var     T ,Pos:Tableau[1..100] de entier ;
        I,J,N,Val :entier ;
Debut
        /*lecture de la taille exacte
        Ecrire('Donner la taille du tableau N≤100') ;
        Repeter Lire(N) Jusqu'à N>0 et N≤100 ;
        Pour I←1 à N Faire Lire(T[I]) ; Fait ;
        Ecrire('Donner Val') ;  Lire(Val) ;
        /*Recherche de val et sa position
        J←0 ;
        Pour I←1 à N Faire
                Si T[I]=Val Alors  J←J+1 ;Pos[J]←I Fsi ;
        Fait ;
        Si J=0    Alors Ecrire(Val,'non trouvée')
                  Sinon Ecrire(Val,'trouvée aux positions :') ;
                        Pour I←1 à J Faire Ecrire (Pos[I]) ; Fait ;
        Fsi ;
        /* Si on initialise J à 1, son incrémentation se fait après l'affectation et la
         /*dimension de Pos devient  J-1
Fin.
```

5- 
```
Algorithme Inverse ;
Var     T :Tableau[1..100] de entier ;
        I,J,X,N:entier ;
Debut
        /*lecture de la taille exacte
        Ecrire('Donner la taille du tableau N≤100') ;
        Repeter Lire(N) Jusqu'à N>0 et N≤100 ;
        /*Lecture des éléments de T
        Pour I←1 à N Faire Lire(T[I]) ; Fait ;
        /*Inverser
        I←1 ; J←N ;
        Tantque  I<J
        Faire
                X←T[I] ; T[I]←T[J]; T[J]←X;
```

```
            I←I+1 ; J←J-1;
        Fait ;
        /*Affichage du nouveau tableau T
        Pour I←1 à N Faire Ecrire(T[I]) ; Fait ;
Fin.
```

6-
```
Algorithme SupprimeZero ;
Var     T :Tableau[1..100] de entier ;
        I,J,N:entier ;
Debut
        /*lecture de la taille exacte
        Ecrire('Donner la taille du tableau N≤100') ;
        Repeter Lire(N) Jusqu'à N>0 et N≤100 ;
        /*Lecture des éléments de T
        Pour I←1 à N Faire Lire(T[I]) ; Fait ;
        /*la suppression des zéro revient à décaler les valeurs non nulles
        I←1 ;
        Tantque  I≤N
        Faire
                Si T[I]=0
                Alors   /*boucle de décalage
                        Pour J←I à N-1
                        Faire T[J]←T[J+1]; Fait ;
                        N←N-1  /*changer la taille du tableau
                Fsi ;
                I←I+1 ;
        Fait ;
        /*Affichage du nouveau tableau T
        Pour I←1 à N Faire Ecrire(T[I]) ; Fait ;
Fin.
```

**Solution 2 :**

```
Algorithme SupprimeZero2 ;
Var     T :Tableau[1..100] de entier ;
        I,J,NBN,N:entier ;
Debut
        /*lecture de la taille exacte
        Ecrire('Donner la taille du tableau N≤100') ;
        Repeter Lire(N) Jusqu'à N>0 et N≤100 ;
        /*Lecture des éléments de T
        Pour I←1 à N Faire Lire(T[I]) ; Fait ;
        /*la suppression des zéro revient à déplacer les valeurs non nulles 1 après 1(sans boucle)
        I←1 ; J←1 ; NBN←0 ;
        Tantque  J≤N
        Faire
                Si T[J]=0
                Alors   NBN←NBN+1  /*nombre de valeurs nulles
                Sinon   /*déplacer l'élément
                        T[I]← T[J] ; I←I+1
                Fsi ;
                J←J+1;
```

```
        Fait ;
        N←N-NBN ;  /*changer la taille de T
        /*Affichage du nouveau tableau T
        Pour I←1 à N Faire Ecrire(T[I]) ; Fait ;
Fin.
```

7-
```
Algorithme NegPuisPos ;
Var     T :Tableau[1..100] de entier ;
        I,J,N,X:entier ;
Debut
        /*lecture de la taille exacte
        Ecrire('Donner la taille du tableau N≤100') ;
        Repeter Lire(N) Jusqu'à N>0 et N≤100 ;
        /*Lecture des éléments de T
        Pour I←1 à N Faire Lire(T[I]) ; Fait ;
        /*passer les valeurs négatives au début
        J←1 ;
        Tantque  J≤N et T[J]<0 Faire J←J+1 ; Fait ;
        /*déplacer les valeurs négatives au début (trouver la première valeur positive I )
        I←J ;
        Tantque  J≤N
        Faire
                Si T[J]<0
                Alors   /*permuter
                        X← T[I] ; T[I]←T[J]; T[J]←X;
                        I←I+1
                Fsi ;
                J←J+1 ;
        Fait ;
        /*Affichage du nouveau tableau T
        Pour I←1 à N Faire Ecrire(T[I]) ; Fait ;
Fin.
```

**EXERCICE 2**

Ecrire un algorithme qui permet d'éclater un vecteur T de N (N≤250) entiers supposés positifs en deux vecteurs T1 et T2 contenant respectivement les nombres pairs et impairs de T.

```
Algorithme Prod ;
Var     T1,T2,T :Tableau[1..250] de entier ;
        I,J,K,N :entier ;
Debut
        /*lecture de la taille exacte
        Ecrire('Donner la taille du tableau N≤250') ;
        Repeter Lire(N) Jusqu'à N>0 et N≤250 ;
        Pour I←1 à N Faire Lire(T[I]) ; Fait ;
        J←1 ;K←1 ; /*compteurs pour les tableaux pairs et impairs
        Pour I←1 à N Faire
                Si T[I] MOD 2=0 Alors  T2[J]← T[I] ; J←J+1
                                Sinon  T1[K]← T[I] ; K←K+1
                Fsi ;
        Fait ;
```

```
        Ecrire('Vecteur pairs') ;  Pour I←1 à J-1 Faire Ecrire(T2[I]) ; Fait ;
        Ecrire('Vecteur impairs') ;  Pour I←1 à K-1 Faire Ecrire (T1[I]) ; Fait ;
        /*taille de T1 et T2 sont respectivement K-1  et  J-1
Fin.
```

**EXERCICE 3**

Soit T un vecteur de N entiers (N≤200). Ecrire un algorithme qui détermine le nombre de succession de deux valeurs (V1 et V2) particulières dans le vecteur.

```
Algorithme Repetition ;
Var     T :Tableau[1..200] de entier ;
        I,N,V1,V2,NBrep :entier ;
Debut
        /*lecture de la taille exacte
        Ecrire('Donner la taille du tableau N≤200') ;
        Repeter Lire(N) Jusqu'à N>0 et N≤200 ;
        Pour I←1 à N Faire Lire(T[I]) ; Fait ;
        Ecrire('Donner deux valeurs V1 et V2') ;
        Lire(V1,V2)
        I←1 ; NBrep←0 ;
        Tantque  I<N
        Faire
                Si T[I]=V1 et T[I+1]=V2
                Alors   NBrep← NBrep+1 ; I←I+2
                Sinon   I←I+1
                Fsi ;
        Fait ;
        Ecrire('Nombre de successions des valeurs',V1,V2,' est :',NBrep) ;
Fin.
```

**EXERCICE 4**

Soient deux vecteurs d'entiers triés V1 (N entiers, N≤100) et V2 (M entiers, M≤150).
Ecrire une procédure qui fusionne ces deux vecteurs dans un autre vecteur V3 trié sans répétition de valeurs identiques.

```
Algorithme Fusion2 ;
Var     A:Tableau[1..100] de entier ; B:Tableau[1..150] de entier ;
        C:Tableau[1..250] de entier ;
        I,J,K,N,M :entier ;
Debut
        Repeter  Lire(N) ; Jusqu'à N>0 et N≤100 ;
        Repeter  Lire(M) ; Jusqu'à M>0 et M≤100 ;
        Pour I←1 à N Faire Lire(A[I]) ; Fait ;
        Pour I←1 à M Faire Lire(B[I]) ; Fait ;
        /*Traitement 1er element
        Si  A[1]<B[1]   Alors   C[1] ←A[1];  I←2; J←1
                Sinon   C[1] ←B[1];  J←2; I←1
        Fsi ;
        K←2 ;
```

```
        Tantque I≤N et J≤M
        Faire
        Si  A[I]<B[J]    Alors    Si C[K]<> C[K-1] Alors C[K] ←A[I];  K←K+1 Fsi; I←I+1
                         Sinon    Si C[K]<> C[K-1] Alors C[K] ←B[J];  K←K+1 Fsi;J←J+1;
        Fsi;
        Fait ;
        Si   I>N Alors  Pour I ←J à M
                         Faire  Si C[K]<> C[K-1] Alors C[K] ←B[I];  K←K+1 Fsi ; Fait ;
        Fsi;
        Si   J>M Alors  Pour J ←I à N
                         Faire  Si C[K]<> C[K-1] Alors C[K] ←A[J];  K←K+1 Fsi;Fait ;
        Fsi;
        Pour I←1 à K-1 Faire Ecrire(C[I]) ; Fait ;
Fin.
```

**EXERCICE 5**

Soit T un tableau de N nombres (N≤50). Ecrire un algorithme qui inverse, dans T, la première séquence croissante de nombres.

```
Algorithme Vecteur ;
Var     T :Tableau[1..50] de entier ;
        I,J,X,N :entier ;
Debut
        Repeter  Lire(N) ; Jusqu'à (N>0) et (N<=50) ;
        //Lecture du vecteur
        Pour I ←1 à N  Faire  Lire(T[I]) ;  Fait ;
        I ←1 ;
        //Position du 1er élément de la séquence (I)
        Tantque (I<N) et(T[I]> T[I+1])  Faire  I ←I+1  Fait ;
        //Position du dernier élément de la séquence  s'il existe (J) et permutation
        Si  I<N   Alors J←I+1;
                         Tantque (J<N) et(T[J]≤ T[J+1])  Faire  ←J+1  Fait ;
                         // Permutation
                         Tantque I<J  Faire
                                  X ← T[I] ;  T[I] ← T[J] ;     T[J] ←X ;
                                  I ←I+1 ;  J ←J-1 ;
                         Fait ;
        Fsi ;
        //Affichage
        Pour I ←1 à N  Faire  Ecrire(T[I]) ;  Fait ;
Fin.
```

**EXERCICE 6**

Soit une matrice A(N, M) de caractères (N≤20 et M≤30). Ecrire un algorithme qui

1- Recherche un élément dans la matrice A.
2- Calcule le nombre de voyelles appartenant à la matrice A.

3- Détermine la transposé de la matrice A.
4- Fait une rotation des colonnes de la matrice A.

```
Algorithme Matrice ;
Var  I,J,N,M,Val,Nbv :entier ; Existe :booleen ;
     A : Tableau[1..20,1..30] de entier;
     AT : Tableau[1..20,1..30] de entier;
Début
        Repeter Lire(N) ; Jusqu'à (N>0) et (N≤20) ;
        Repeter Lire(M) ; Jusqu'à (M>0) et (M≤30) ;
        //Lecture de la Matrice
        Pour  I ←1 à N  Faire  Pour  J ←1 à M  Faire
                                                Lire(A[I,J]) ;
                                        Fait ;
                        Fait ;
        Ecrire('Donner Val') ;  Lire(Val) ;
        /*Recherche de Val
        Existe←Faux ; I←1 ;
        Tanque  I ≤N et Non Existe
        Faire   J←1 ;
                Tanque  J ≤M et Non Existe
                Faire
                        Si A[I,J]=Val    Alors Existe←Vrai Fsi ;
                        J←J+1 ;
                Fait ;
                I←I+1 ;
        Fait ;
        Si Existe Alors Ecrire(Val,'Existe') Sinon Ecrire(Val,'Non trouvée') Fsi ;
        /* nombre de voyelles dans A
        Nbv←0 ;
        Pour  I ←1 à N  Faire  Pour  J ←1 à M  Faire
                                                Cas A[I,J] Vaut
                                        'a','e','i','o','u','y' : Nbv←Nbv+1 ;
                                        Fincas ;
                                Fait ;
                        Fait ;
        Ecrire('Nombre de voyelles est :',Nbv) ;
Fin.
```

**EXERCICE 7**

Soit une matrice carrée A(N, N) d'entiers (N≤25). Ecrire deux des actions paramétrées permettant de :
1- Calculer la trace de la matrice A. (*La trace est la somme des éléments de la diagonale principale).*
2- Déterminer le maximum et sa position, des valeurs des deux diagonales (principale et secondaire).

```
Algorithme Trace ;
Var  I,J,N,Max,Lmax,Cmax,Tr :entier ;    A : Tableau[1..25,1..25] de entier;
Début
```

```
        Repeter Lire(N) ; Jusqu'à (N>0) et (N≤25) ;
        //Lecture de la Matrice
        Pour I ←1 à N Faire Pour J ←1 à N Faire Lire(A[I,J]) ; Fait ; Fait ;
        // et calcul de la trace
        Tr←0 ;
        Pour I ←1 à N Faire Tr←Tr+ A[I,I]) ; Fait ;
        //Max et sa position
        Max←A[1,1] ;
        Pour I ←1 à N
        Faire Si Max< A[I,I]) Alors Max← A[I,I]; Cmax←I;Lmax←I Fsi ; /*Diag Princ
              /*Diag secondaire – relation renter indice   I------- N+1-I
              Si Max< A[I,N+1-I]) Alors Max← A[I,N+1-I]; Cmax←N+1-I;Lmax←I Fsi;
        Fait ;
        Ecrire('Max=',Max,'Position Ligne :',Lmax,' Colonne :',Cmax) ;
Fin.
```

**EXERCICE 8**

Soit une matrice A(N, M) d'entiers (N≤20 et M≤30), écrire un algorithme qui :

- Calcule et sauvegarde la somme de chaque colonne,
- Détermine la position Jmin de la somme minimale et la position Jmax de la somme maximale.
- Permute les deux colonnes d'indices Jmin et Jmax de la matrice A si Jmin > Jmax.

```
Algorithme Matrice ;
Var I,J,N,M,Jmin,Jmax,X :entier ;
    A : Tableau[1..20,1..30] de entier;   Som : Tableau[1..30] de entie
Début
        Repeter Lire(N) ; Jusqu'à (N>0) et (N≤20) ;
        Repeter Lire(M) ; Jusqu'à (M>0) et (M≤30) ;
        //Lecture de la Matrice
        Pour I ←1 à N Faire Pour J ←1 à M Faire Lire(A[I,J]) ; Fait ; Fait ;
        //calcul et save de la somme de chaque col
        Pour J ←1 à M
        Faire   Som[J]←0 ;
                Pour I ←1 à N
                Faire
                        Som[J]← Som[J]+ A[I,J];
                Fait ;
         Fait ;
        Pour J ←1 à M Faire Ecrire(Som[J]) Fait ;
        Jmin← Som[1] ; Jmax← Som[1] ;
        Pour J ←1 à M
        Faire   Si Som[Jmax] < Som[J] Alors Jmax←J Fsi ;
                Si Som[Jmin] > Som[J] Alors Jmin←J Fsi ;
        Fait ;
        Ecrire('Position Jmin=',Jmin,' Jmax=',Jmax) ;
        Si Jmin>Jmax
        Alors   Pour I ←1 à N
```

```
            Faire  X← A[I,Jmin]; A[I,Jmin]← A[I,Jmax]; A[I,Jmax]←X Fait;
            //affichage
            Pour  I ←1 à N  Faire  Pour  J ←1 à M  Faire  Ecrire(A[I,J]) ; Fait ; Fait ;

      Fsi;
Fin.
```

**EXERCICE 9**

1- Ecrire une AP qui vérifie si deux vecteurs sont identiques ;
2- Ecrire un algorithme permettant de supprimer toutes les colonnes identiques d'une matrice. A[N,M].

On déclare un **Type** Vect dans la clause **Type**
**Type** Vect=Tableau[1..100] de entier ;

1- Fonction Identique(V1,V2 :vect ; N,M :entier) :booleen ;

```
Var  I :entier ; Idem :booleen ;
Debut
        Si N<>M
        Alors Idem←Faux
        Sinon   Idem←Vrai ; I←1 ;
        Tantque I≤N et Tdem Faire Si V1[I]<> V2[I] Alors Idem←Faux Fsi ;
                                   Fait
        Fsi ;
        Identique←Idem ;
Fin ;
```

2- **Algorithme** SupIdem ;

```
Type    Vect=Tableau[1..100] de entier ;
        Matrice=Tableau[1..100,1..150] de entier ;
Var     V1,V2 :Vect ; A :Matrice ;
I,J,K,N,M:entier ;
        Fonction Identique(V1,V2 :vect ; N,M :entier) :booleen ;
---------- reprendre les actions de la fonction ------------
Debut
Repeter Lire(N) ; Jusqu'à (N>0) et (N≤100) ;
Repeter Lire(M) ; Jusqu'à (M>0) et (M≤150) ;
//Lecture de la Matrice
Pour  I ←1 à N  Faire  Pour  J ←1 à M  Faire  Lire(A[I,J]) ; Fait ; Fait ;

/*Lecture du vecteur
Ecrire('Donner un vecteur de taille N') ;
Pour  I ←1 à N  Faire Lire(V1[I]) ; Fait ;

/* localiser les colonnes identiques à V1 et les supprimer
J ←1 ;
Tantque  J≤ M
Faire
/*récupérer la colonne dans V2
Pour  I ←1 à N  Faire V2[I]←A[I,J] ; Fait ;
```

```
Si Identique(V1,V2, N,N)  /*les deux vecteurs ont la même dim.
Alors /*suppression de la colonne J qui revient à décaler les colonnes de droite.
        Pour  I ←1 à N
                Faire   Pour K←J à M-1
                        Faire A[I,K] ← A[I,K+1] ;
                        Fait ;
                Fait ;
                M←M-1  /*diminuer le nombre de colonnes sans passer à la colonne suiv, car la
                  /* colonne décalée peut être aussi identique à V1
        Sinon J←J+1
        Fsi ;
Fait ;
```

**EXERCICE 10**

1- Ecrire une action paramétrée qui détermine la présence ou non d'un caractère dans une chaîne.
2- Ecrire une action paramétrée qui comptabilise le nombre de voyelle dans une chaîne.
3- Ecrire une action paramétrée qui détermine si une phrase donnée contient toutes les voyelles.

1-
```
Fonction Presence(CH :chaine[200] ;C :caractère):booleen ;
Var  T,I :entier ; Pr :booleen ;
Debut
        T←Taille(CH) ;  Pr←Faux ;  I←1 ;
        Tantque I≤T et Non Pr  Faire Si CH[I]=C Alors Pr←Vrai Fsi ; I←I+1 ; Fait ;
        Presence ←Pr ;
Fin ;
```

2-
```
Fonction NBVoyelle(CH :chaine[200] ):entier ;
Var  T,I,Nbv :entier ;
Debut
        T←Taille(CH) ;  Nbv←0 ;
        Pour  I ←1 à T
                Faire  Cas CH[I] Vaut
                'a','e','i','o','u','y' : Nbv←Nbv+1 ;
                Fincas ;
        Fait ;
        NBVoyelle ←Nbv ;
Fin ;
```

3-
```
Fonction TouVoyelle(PH :chaine[200] ):booleen ;
Var  T,I :entier ;  CV :chaine[6] ;
Debut
        T←Taille(PH) ;  CV←'aeiouy' ;
        Pour  I ←1 à T
        Faire  Cas PH[I] Vaut
                'a' :CV[1]←'X' ;
                'e' : CV[2]←'X' ;
                'i' : CV[3]←'X' ;
                'o' : CV[4]←'X' ;
                'u' : CV[5]←'X' ;
                'y' : CV[6]←'X' ;
                Fincas ;
```

```
        Fait ;
        Si CV='XXXXXX' Alors TouVoyelle ←Vrai Sinon  TouVoyelle ←Faux Fsi;
Fin ;
```

**EXERCICE 11**

1- Ecrire une fonction qui vérifie l'existence d'une sous-chaîne dans une chaîne.
2- Ecrire une action paramétrée qui supprime (élimine) la suite de N caractères de la chaîne CH à partir de la position P.
3- Ecrire une fonction qui détermine si un mot est un palindrome. (*Un mot palindrome se lit de gauche à droite et de droite à gauche (ex : RADAR, ELLE, ICI)*).

1- Fonction SHexiste(SH ,CH :chaine[250]) :booleen ;

```
Var T,Ts,I :entier ;  EX :booleen ;
Debut
        T←Taille(CH) ; Ts←Taille(SH) ;  EX←Faux ;
        Si T≥Ts
        Alors   I←1 ;
        Tantque I≤(T-Ts+1) et Non EX
        Faire   J←I ; K←1 ; EX←Vrai ;
                Tantque K≤Ts et EX
                Faire   Si CH[J]<>SH[K] Alors EX←Faux Fsi ;
                        J←J+1 ;  K←K+1 ;
                Fait ;
                I←I+1 ;
        Fait ;
        Fsi ;
        SHexiste←EX ;
Fin ;
```

2- Procedure SupprimeNP(E/S/ CH :chaine[250], E/ N,P:entier) ;

```
Var T,I :entier ;  CHI:chaine[250] ;
Debut
        T←Taille(CH) ;
        Si P≤T
        Alors   CHI←'' ;  /*initialiser à vide
                Pour  I ←1 à P-1 Faire CHI[I]←CH[I] ; Fait ; /*copier la partie avant P
                /*Ajouter les caractères après les N à supprimer
                Si (P+N)>T Alors N←T-P  Fsi ; /* s'il reste moins de N caractères, on sup. tous
                Pour    I←P+N à T  Faire CHI←CHI+CH[I] ; Fait ;
                CH←CHI ;
        Fsi ;
Fin ;
```

3- Fonction Palindrome (CH :chaine[250]) :booleen ;

```
Var I,J :entier ;  Pal :booleen ;
Debut
        I←1 ; J←Taille(CH) ;  Pal←Vrai ;
        Tantque I<J et Pal
        Faire
                Si CH[I]<>CH[J] Alors Pal←Faux Fsi ;
                I←I+1 ;  J←J-1 ;
        Fait ;
        Palindrome←Pal;
Fin ;
```

**EXERCICE 12**

Ecrire un algorithme qui comptabilise le nombre de caractères, de mots et de phrases dans un texte. Les mots sont séparés par des espaces et les phrases séparées par un point. (Sans compter les séparateurs : espace et point)

```
Algorithme Texte ;
Var TX :chaine ;
     T,NbC,NbM,NbP,I :entier ;
Debut
   Ecrire('Donner un texte') ; Lire(TX) ;
   T←Taille(TX) ;
   NbC←0 ; NbM←0 ; NbP←0 ; I←1 ;
   Tantque I≤N
   Faire Si TX[I]=' '
         Alors I←I+1
         Sinon Tantque (I≤N)et(TX[I]≠' ')et(TX[I]≠'. ')
               Faire NbC←NbC+1 ; I←I+1 ; Fait ;
               NbM←NbM+1 ;
               Si TX[I]='. ' Alors NbP←NbP+1 Fsi ;
         Fsi ;
   Fait ;
   Ecrire('Nombre de caractère :',NbC,' Nombre de mots :',NbM,' Nombre de phrases :',NbP) ;
Fin.
```

**EXERCICE 13**

Ecrire un algorithme qui lit deux mots et qui détermine s'ils sont anagrammes. Sachant qu'un mot est dit anagramme d'un autre mot s'ils utilisent (sont formés par) les même lettres.

Exemples :
CHIEN anagramme de CHINE, NICHE,
AIMER anagramme de MAIRE, MARIE, RAMIE,
GELER n'est pas anagramme d' ALGER, …

- **Première solution** : on trie les deux mots puis on compare

```
Algorithme Anagramme ;
Var M1,M2 :chaine ;
    Procedure TRI(E/S/ CH :chaine) ;
    Var I,J,T:entire; X :caractère ;
    Debut
       T←Taille(CH) ;
       Pour I←1 à T-1
       Faire  Pour J←I+1 à T
              Faire Si CH[I]>CH[J]
                    Alors    X←CH[I] ; CH[I]←CH[J]; CH[J]←X;
```

```
                    Fsi;
                Fait;
            Fait;
        Fin;
Debut
    Ecrire('donner 2 mots') ;
    Lire(M1,M2) ;
    TRI(M1) ; TRI(M2) ;
    Si M1=M2 Alors Ecrire('Les deux mots sont anagrammes')
                Sinon Ecrire('Les deux mots ne sont pas anagrammes')
    Fsi ;
Fin.
```

**- Deuxième solution** : les deux mots puis sont anagrammes si chaque caractère apparait le même nombre de fois dans les deux mots.

```
Algorithme Anagramme ;
Var M1,M2 :chaine ;
    T1,T2,I,F1,F2 :entier ;
    X :caractère ;
    Anag :booleen ;
Debut
    Ecrire('donner 2 mots') ;
    Lire(M1,M2) ;
    T1←Taille(M1) ; T2←Taille(M2) ;
    Anag←Faux ;
    Si T1=T2
    Alors Anag←Vrai ; I←1 ;
        Tantque I≤T1 et Anag
        Faire X←M1[I] ; F1←0 ;
            Pour J←1 à T1 Faire Si M1[J]=X Alors F1←F1+1 Fsi ; Fait ;
            F2←0 ;
            Pour J←1 à T1 Faire Si M2[J]=X Alors F2←F2+1 Fsi ; Fait ;
            Si F1≠F2 Alors Anag←Faux Fsi ;
            I←I+1 ;
        Fait ;
    Fsi ;
    Si Anag    Alors Ecrire('Les deux mots sont anagrammes')
            Sinon Ecrire('Les deux mots ne sont pas anagrammes')
    Fsi ;
Fin.
```

**- Troisième solution** : chaque caractère de M1 trouvé dans M2 est remplacé par un car spécial ( ex : '*'). A la fin on vérifie si tous las caractères de M2 ont été remplacés.

```
Algorithme Anagramme ;
Var M1,M2 :chaine ;
```

```
    T1,T2,I,J :entier ;
    Anag :booleen ;
Debut
    Ecrire('donner 2 mots') ;
    Lire(M1,M2) ;
    T1←Taille(M1) ; T2←Taille(M2) ;
    Anag←Faux ;
    Si T1=T2
    Alors Anag←Vrai ; I←1 ;
          Tantque I≤T1 et Anag
          Faire J←1 ; Anag←Faux ;
                Tanque J≤T1 et Non Anag
                Faire Si M1[I]=M2[J] Alors M2[J]←'*' ; Anag←Vrai Fsi ; J←J+1 ; Fait ;
                I←I+1 ;
          Fait ;
          I←1 ;
          Tantque I≤T1 et Anag
          Faire Si M2[I]≠'*' Alors Anag←Faux Fsi ; I←I+1 ; Fait ;
    Fsi ;
    Si Anag    Alors Ecrire('Les deux mots sont anagrammes')
               Sinon Ecrire('Les deux mots ne sont pas anagrammes')
    Fsi ;
Fin.
```

**EXERCICE 1**

1- Définir un type TEMPS qui contient les champs heure, minute, seconde.
2- Ecrire une action paramétrée qui réalise la somme T de deux durées T1 et T2 de type temps.
3- Ecrire une fonction TRANSFORM qui transforme un temps T de type TEMPS en un entier S qui exprime ce temps en secondes.
   Exemple : pour T = 2 heures 10 minutes 37 secondes, S = 7837 secondes.
4- Ecrire une procédure DECOMPOS qui décompose un temps S exprimé en secondes en un temps T de type TEMPS.
   **Exemple** : pour S = 7837 secondes, T = 2 heures 10 minutes 37 secondes.
5- Etant donnés deux temps T1 et T2 de type TEMPS, écrire un algorithme qui calcule le temps T somme des temps T1 et T2 (T, T1 et T2 sont de type TEMPS) en utilisant les actions TRANSFORM et DECOMPOS.

1- **Type** TEMPS=Enregistrement H,M,S :entier ; Fin ;
2- **Procedure** SommeT(E/ T1,T2 :TEMPS ; S/ T :TEMPS) ;
**Var** X :entier ;
**Debut**
X←T1.S+T2.S;
T.S←X mod 60; T.M← X div 60;
X←T.M+T1.M+T2.M;
T.M← X mod 60; T.H← X div 60+ T1.H+T2.H;
**Fin** ;
3- **Fonction** TRANSFORM(T :TEMPS) :entier ;
**Debut**
TRANSFORM←T.S+60*T.M+3600*T.H ;
**Fin** ;
En algorithmique, on ne peut pas avoir une fonction de type enregistrement, donc on utilise une procédure.
4- **Procedure** DECOMPOS(E/ S :entier ; S/ T :TEMPS) ;
**Debut**
T.H←S div 3600 ; S← S mod 3600 ;
T.M←S div 60 ; T.S← S mod 60 ;
**Fin**;
5-
**Algorithme** CalculT;
**Type** TEMPS=Enregistrement H,M,S :entier ; Fin ;
**Var** T1,T2,T :TEMPS ;
S :entier ;
**Debut**
Ecrire('Donner un Temps T1 : H M S') ;
Lire(T1.H,T1.M,T1.S) ;
Ecrire('Donner un Temps T2 : H M S') ;
*/*on peut lire en utilisant l'instruction Avec*
**Avec** T2 **Faire** Lire(H,M,S) ; **Fait** ;
*/*transformer T1 et T2 en secondes, puis additionner*
S← TRANSFORM(T1) + TRANSFORM(T2) ;
*/*décomposer S en TEMPS T*

```
    DECOMPOS(S,T) ;
    /*On peut faire aussi DECOMPOS(TRANSFORM(T1) + TRANSFORM(T2) ,T) ;
    Ecrire('La somme est :',T.H,' :',T.M,' :',T.S) ;
Fin.
```

**EXERCICE 2**

Un nombre complexe Z est entièrement défini par ses parties réelle **a** et imaginaire **b** ( **Z = a + bi** ).

1- Donner la déclaration d'un nombre complexe,
2- Ecrire les fonctions : **ReelZ**, **ImagZ** et **Module** donnant les attributs d'un nombre complexe respectivement : la partie réelle, la partie imaginaire et le module),
3- Ecrire les actions paramétrées : **SommeZ**, **DiffZ** et **ProdZ** nécessaires à l'arithmétique sur les complexes, respectivement pour l'addition, la soustraction et la multiplication,
4- Ecrire une procédure **ConjZ** qui calcule le conjugué d'un nombre complexe.
5- Ecrire une fonction **EgaleZ** qui teste l'égalité de deux nombres complexes.
6- Ecrire une procédure **EcrireZ** qui permet d'afficher un nombre complexe.

Soit TC un tableau de N nombres complexes (N<=100). En utilisant les actions paramétrés précédentes, écrire un algorithme qui :
- Affiche l'élément de **TC** ayant le plus grand module. Puis vérifie l'existence de son conjugué dans **TC**.
- Calcule la somme **Zs** et le produit **Zp** des éléments non nuls du tableau **TC**.
- Calcule et affiche la différence entre **Zs** et **Zp** si elle est imaginaire pur.

```
1- Type Tcomplexe=Enregistrement a,b :reel ; Fin ;
2- Fonction ReelZ(Z :Tcomplexe) :reel ;
   Debut ReelZ←Z.a ; Fin ;

   Fonction ImagZ(Z :Tcomplexe) :reel ;
   Debut ImagZ←Z.b ; Fin ;

   Fonction ModuleZ(Z :Tcomplexe) :reel ;
   Debut ModuleZ←Racine(Z.a*Z.a+Z.b*Z.b) ; Fin ;


3- Procedure SommeZ(E/ Z1,Z2 :Tcomplexe ; S/ Z :Tcomplexe) ;
   Debut
       Z.a←Z1.a+Z2.a ; Z.b←Z1.b+Z2.b ;
   Fin ;

   Procedure DiffZ(E/ Z1,Z2 :Tcomplexe ; S/ Z :Tcomplexe) ;
   Debut
       Z.a←Z1.a-Z2.a ; Z.b←Z1.b-Z2.b ;
   Fin ;

   Procedure ProdZ(E/ Z1,Z2 :Tcomplexe ; S/ Z :Tcomplexe) ;
   Debut
       Z.a←Z1.a*Z2.a- Z1.b*Z2.b ;
       Z.b← Z1.a*Z2.b+ Z1.b*Z2.a ;
   Fin ;

4- Procedure ConjZ(E/ Z :Tcomplexe ; S/ ZC :Tcomplexe) ;
```

```
   Debut
        ZC.a←Z.a ; ZC.b← -Z.b;
   Fin ;

5- Fonction EgaleZ (Z1,Z2 :Tcomplexe) :Booleen ;
   Debut
        Si Z1.a=Z2.a et Z1.b=Z2.b Alors EgaleZ← Vrai Sinon EgaleZ← Faux Fsi ;
   Fin ;

6- Procédure EcrireZ (Z:Tcomplexe) ;
   Debut
        Si Z.b=0  AlorsEcrire(Z.a)
                  Sinon Si Z.a=0   Alors Ecrire(Z.b,' i')
                                   Sinon Si Z.b>0 Alors Ecrire(Z.a,'+',Z.b,' i')
                                                  Sinon Ecrire(Z.a,Z.b,' i')
                                         Fsi ;
                        Fsi ;
        Fsi ;
   Fin ;
```

7-

```
Algorithme Complexe ;
Type Tcomplexe=Enregistrement a,b :reel ; Fin ;
Const  Z0.a=0 ; Z0.b=0 ; /*decalation d'une constante complexe nulle ;
Var TC :Tableau[1..100] de Tcomplexe ;
      I,N :entier ;
      Z,Zs,Zp,Zm :Tcomplexe ;
      M,Max :reel ;
      Trouve :booleen ;
      /*declaration des différentes APs
        -   -   -   -
Debut

        Ecrire('Donner N') ;
        Repeter Lire(N) ; Jusqu'à N>0 et N≤100 ;
        /*lecture du tableau des complexes
        Pour I←1 à N
        Faire Lire(TC[I].a, TC[I].b) ;
              /* on peut faire aussi
                Avec T[I] Faire Lire(a,b) Fait ;
             /*ou encore utiliser une variable intermédiaire de type Tcomplexe
                Lire(Z.a,Z.b) ;
                T[I]←Z;
        Fait ;
        /*Recherche du nombre ayant le plus grand module
        Zm←TC[1] ; Max←ModuleZ(Zm) ;
```

```
        Pour I←2 à N
        Faire
                M← ModuleZ(TC[I]) ;
                Si M>Max Alors      Zm← TC[I] ;
                                    Max←M
                Fsi ;
        Fait ;
        Ecrire('Le nombre complexe ayant le plus grand module est :',EcrireZ(Zm)) ;
        /* Recherche du conjugué de Zm dans TC
        Trouve←Faux ; I←1 ;
        Tantque I≤N et Non Trouve
        Faire
                Trouve← EgaleZ(TC[I],ConjZ(Zm)) ;
                I←I+1 ;
        Fait ;
        Si Trouve Alors Ecrire('Len conjugué existe' Sinon Ecrire('le conjugué n''existe pas') Fsi ;
        /*calcule de la somme et du produit, on initialise Zs à Z0 et Zp à 1
        Zs←Z0 ; Zp.a←1 ; Zp.b←0 ;
        Pour I←1 à N
        Faire
            Si Non EgaleZ(TC[I],Z0)
            Alors
                SommeZ(Zs,TC[I],Zs) ;
                ProdZ(Zp,TC[I],Zp)
            Fsi;
        Fait ;
        DiffZ(Zs,Zp,Z) ;
        Si ReelZ(Z)=0 et ImagZ(Z)≠0 Alors EcrireZ(Z) Fsi ;
Fin.
```

**EXERCICE 3**

Soit Tdate un type date composé des champs entiers JJ,MM,AA.

- Ecrire une AP **CompareD** permettant de comparer deux dates D1 et D2.
- Soit TD un tableau de N dates (N≤100). En utilisant l'AP **CompareD,** écrire un algorithme permettant de trier ce tableau dans l'ordre croissant des dates.

```
Type Tdate = Enregistrement JJ,MM,AA :entier ; Fin ;
On considère une Fonction pouvant prendre 1 Pour >, 0 Pour = et -1 Pour <
Fonction CompareD(D1,D2 :Tdate) :entier ;
Debut
        Si D1.AA>D2.AA
```

```
        Alors CompareD←1
        Sinon  Si D1.AA<D2.AA
                Alors CompareD← -1
                Sinon  Si D1.MM>D2.MM
                        Alors  CompareD←1
                        Sinon  Si D1.MM<D2.MM
                                Alors CompareD← -1
                                Sinon  Si D1.JJ>D2.JJ
                                        Alors CompareD←1
                                        Sinon  Si D1.JJ<D2.JJ
                                                Alors CompareD← -1
                                                Sinon CompareD← 0
                                                Fsi
                                        Fsi
                                Fsi
                        Fsi
                Fsi
        Fsi ;
Fin ;

Algorithme TriDate ;
Type Tdate = Enregistrement JJ,MM,AA :entier ; Fin ;
Var TD :Tableau[1..100] de Tdate ;
    D :Tdate ;
    I,J :entier ;
     Fonction CompareD(D1,D2 :Tdate) :entier ;
       -   - - -
Debut
        Ecrire('Donner N') ;
        Repeter Lire(N) ; Jusqu'à N>0 et N≤100 ;
        /*lecture du tableau des dates
        Pour I←1 à N
        Faire Lire(TD[I].JJ, TD[I].MM, TD[I].AA); Fait;
        /*le tri
        Pour I←1 à N-1
        Faire
                Pour J←I+1 à N
                Faire
                        Si CompareD( TD[I], TD[J])=1
                        Alors  D← TD[I]; TD[I]←TD[J]; TD[J]←D
                        Fsi;
                Fait;
        Fait;
```

```
        /*affichage des dates triées
        Pour I←1 à N
        Faire  Ecrire(TD[I].JJ,'/', TD[I].MM,'/', TD[I].AA); Fait;
Fin.
```

**EXERCICE 4**

Ecrire une fonction qui détermine la différence en nombre de jours entre deux dates.

**Type** Tdate = Enregistrement JJ,MM,AA :entier ; **Fin** ;
Nous allons écrire quelques fonctions utiles :

- Une fonction qui détermine si une année est bissextile :

```
Fonction BIS6(A :entier) :booleen ;
Debut
        Si (A mod 4=0 et A mod 100≠0)ou(A mod 400=0)
        Alors BIS6←Vrai Sinon BIS6← Faux
        Fsi ;
Fin ;
```

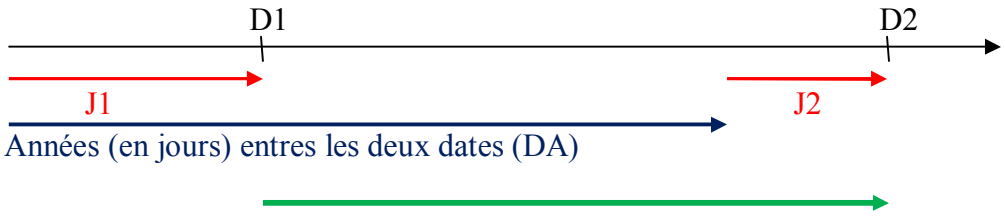
- Une fonction NBJour qui donne le nombre de jours du 01/01/AA à JJ/MM/AA d'une date donnée

```
Fonction NBjour(D :Tdate) :entier ;
Var I :entier ;
Debut
        NBJour←D.JJ ;
        Pour I←1 à D.MM-1
        Faire
                Cas I Vaut
                1,3,5,7,8,10,12 : NBJour←NBJour+31 ; /*mois Avec 31 jours
                4,6,9,11 : NBJour←NBJour+30 ; /*mois Avec 30 jours
                2 : Si BIS6(D.AA) Alors NBJour←NBJour+29 Sinon NBJour←NBJour+28 Fsi ;
                Fincas ;
        Fait ;
Fin ;
```

On reprend la **Fonction** qui compare 2 dates.

```
Fonction CompareD(D1,D2 :Tdate) :entier ;
        -   - - - -
```

Enfin notre fonction :



**Différence (en jours) entres les deux dates = J2+DA-J1**

```
Fonction DiffJour(D1,D2 :Tdate) :entier ;
Var J1,J2 :entier ;
    D :Tdate ;
Debut
      /*trier les deux dates
      Si CompareD(D1,D2)=1 Alors D←D1 ; D1←D2 ; D2←D Fsi ;
      /*Compter le nombre de jours de D1 et celui de D2
      J1←NBJour(D1) ; J2←NBJour(D2) ;
      /*Ajouter les années (en jours) entres les deux dates
      Pour I←D1.AA à D2.AA-1
      Faire Si BIS6(I) Alors J2←J2+366 Sinon J2←J2+365 Fsi ; Fait ;
      DiffJour←J2-J1 ;
Fin ;
```

**EXERCICE 5**

Soit un enregistrement E défini par deux informations :
- T un tableau d'entiers pouvant contenir au maximum 100 éléments;
- N le nombre d'éléments du tableau T.

Soit une chaîne de caractères M, écrire une action paramétrée qui retourne un enregistrement de type E contenant toutes les positions de la chaîne 'ab' dans la chaîne M.

Exemple : M = 'faabaababbaabrs'
Positions : 3 - 6 - 8 - 12
Nombre d'éléments : 4

```
Type E=Enregistrement
                T :Tableau[1..100] de entier ;
                N :entier ;
          Fin ;
Procedure ABPos(E/ M :chaine[250] ;S/ Pos :E) ;
Var I,J,T :entier ;
Debut
      T←Taille(M) ; I←1 ; J←1 ;  S.N←0 ;
      Tantque I<T
      Faire  Si M[I]='a' et M[I+1]='b'
             Alors S.T[J] ←I ; S.N←S.N+1 ; J←J+1 ; I←I+2
             Sinon I←I+1
             Fsi ;
      Fait ;
Fin ;
```

**EXERCICE 6**

Considérons les types d'enregistrements suivants :

```
Type  TDate = Enregistrement
                    Jour, mois, année : entier ;
```

```
            Fin;
        TAdresse = Enregistrement
                                Numéro : entier ;
                                Rue : chaine [50] ;
                                Ville : chaine [20] ;
                                Wilaya : chaine [20] ;
                                Cw : entier ;          { Code Wilaya }
                        Fin;
        THabitant = Enregistrement
                                Nom, prenom : chaine [20] ;
                                Date_naiss : date ;
                                Residence : Adresse ;
                        Fin;
```

Ecrire un algorithme permettant de :

1- Remplir un tableau T de N habitants (N≤100).
2- Afficher à partir de T les adresses des habitants nés avant une année de naissance donnée.
3- Afficher les noms et les dates de naissance des habitants de la ville de 'Zemmouri' de la wilaya de 'Boumerdes'.
4- Editer le nombre d'habitants par wilaya.

```
Algorithme Habitant ;
Type  TDate = Enregistrement
                        Jour, mois, année : entier ;
              Fin;
        TAdresse = Enregistrement
                                Numéro : entier ;
                                Rue : chaine [50] ;
                                Ville : chaine [20] ;
                                Wilaya : chaine [20] ;
                                Cw : entier ;          { Code Wilaya }
                        Fin;
        THabitant = Enregistrement
                                Nom, prenom : chaine [20] ;
                                Date_naiss : date ;
                                ReSidence : Adresse ;
                        Fin;
Var     T :Tableau[1..100] de THabitant ;
        TW :Tableau[1..48] de entier ;
        I,N,A :entier ;
        EH :THabitant ;
Debut
        Ecrire('Donner N') ;
        Repeter Lire(N) ; Jusqu'à N>0 et N≤100 ;
```

```
/*lecture du tableau des Habitants
Pour I←1 à N
Faire
        Avec EH,EH.Date_naiss,EH.Residence
        Faire
                Lire(Nom,Prenom) ;
                Lire(Jour,Mois,Annee) ;
                Lire(Numero,Rue,Ville,Wilaya,CW) ;
        Fait ;
        T[I]←EH;
Fait;
/*affichage des adresses des habitants nés avant une année A
Ecrire('Donner une Année') ; Lire(A) ;
Pour I←1 à N
Faire
        Avec T[I].Date_naiss, T[I].Residence
        Faire
                Si Annee=A Alors Ecriree(Numero,' ',Rue,'  ',Ville,' ',Wilaya) Fsi;
        Fait ;
Fait;
/*affichage des noms et adresses des habitants de zemmouri
Pour I←1 à N
Faire
        Avec T[I],T[I].Date_naiss,T[I].Residence
        Faire
                Si Ville='zemmouri' et Wilaya='Boumerdes'
                Alors Ecriree(Nom,Prenom,'  ',Jour,'/',Mois,'/',Annee) Fsi;
        Fait ;
Fait;
/*nombre d'habitants par wilaya ;
/*initialiser à 0 ;
Pour I←1 à 48 Faire TW[I]←0 ; Fait ;
Pour I←1 à N
Faire TW[T[I].ReSidence.CW]← TW[T[I].Residence.CW]+1 ;Fait ;
        /*on peut aussi utiliser une variable intermédiaire
        /* A← T[I].Residence.CW; TW[A]← TW[A]+1;
/*affichage
Pour I←1 à 48  Faire Ecrire('Wilaya ',I,'   Nombre  ',TW[I]) ; Fait ;
Fin.
```

**EXERCICE 7**

On s'intéresse à la gestion des véhicules d'un parc auto. Chaque véhicule est caractérisé par un matricule, une marque, un modèle, une couleur, le nombre de places, une puissance fiscale.

1- Donner l'enregistrement permettant de décrire un véhicule.
2- Décomposer le matricule en ses composants élémentaires puis donner la nouvelle structure de l'enregistrement.
3- Ecrire un algorithme qui permet de :
- Stocker les informations d'un parc auto regroupent au max 50 véhicules en utilisant les structures adéquates ;
- Etablir la liste (matricule, marque, modèle, puissance) des véhicules d'une couleur donnée ;
- Etablir un tableau statistique contenant le nombre de véhicules immatriculés par wilaya ;

1-

```
Type   TVehicule=Enregistrement
                    Matricule :chaine[11];
                    Marque :chaine[20];
                    Modele :chaine[10] ;
                    Nbp,Puis :entier ;
                  Fin ;
```

2- Nouvelle structure après décomposition du matricule :

```
Type   TMatricule=Enregistrement
                        Num :chaine[6] ;
                        Cat :caractère ;
                        Annee :chaine[2] ;
                  Fin ;
       TVehicule=Enregistrement
                    Matricule :TMatricule;
                    Marque :chaine[20];
                    Modele,Couleur :chaine[10] ;
                    Nbp,Puis :entier ;
                  Fin ;
```

3-

```
Algorithme Parc ;
Type   TMatricule=Enregistrement
                        /*déclarer les différents champs comme chaine afin de pouvoir afficher les 0
                        /*sans faire de traitements
                        Num :chaine[6] ;
                        Cat :caractère ;
                        Annee :chaine[2] ;
                        Cw :chaine[2] ;
                  Fin ;
       TVehicule=Enregistrement
                    Matricule :TMatricule;
                    Marque :chaine[20];
                    Modele,Couleur :chaine[10] ;
                    Nbp,Puis :entier ;
```

```
                Fin ;
Var   TP :Tableau[1..500] de Tvehicule ;
      TW :Tableau[1..48] de entier ;
      V :TVehicule ;
      I,N,A :entier ;
      Col :chaine[10] ;
      /*Fonction permettant de convertir une chaine en un entier ;
      FonctionValCh(ch :chaine[10]) :entier ;
      Var I,T,P,V :entier ;
      Debut
            P←1 ; ValCh←0 ; T←Taille(ch) ;
            Pour I←1 à T
            Faire  Cas ch[T-I+1] Vaut
                  '0' :V←0 ;
                  '1' :V←1 ;
                  '2' :V←2 ;
                  '3' :V←3 ;
                  '4' :V←4 ;
                  '5' :V←5 ;
                  '6' :V←6 ;
                  '7' :V←7 ;
                  '8' :V←8 ;
                  '9' :V←9 ;
                  Fincas ;
                  ValCh←ValCh+V*P ;
                  P←P*10;
            Fin;
Debut
      Ecrire('Donner N') ;
      Repeter Lire(N) ; Jusqu'à N>0 et N≤500 ;
      /*lecture du tableau des Véhicules
      Pour I←1 à N
      Faire
            Avec V,V.Matricule
            Faire
                  Lire(Num,Cat,Annee) ;
                  Lire(Marque,Modele,Couleur,Nbp,Puis) ;
            Fait ;
            TP[I]←V;
      Fait;
      Ecrire('Donner une couleur') ; Lire(Col) ;
      Pour I←1 à N
      Faire
```

```
            Avec TP[I], TP[I].Matricule
            Faire
                  Si Col=Couleur
                  Alors
                        Ecrire(Num,'-',Cat+Annee,'-',Cw) ;
                        Ecrire(Marque,Modele, ,Puis) ;
                  Fsi ;
            Fait ;
      Fait;
      /*nombre de véhicules par wilaya ;
      /*initialiser à 0 ;
      Pour I←1 à 48 Faire TW[I]←0 ; Fait ;
      Pour I←1 à N
      Faire A← ValCh(TP[I].Matricule.Cw); TW[A]← TW[A]+1; Fait;
      /*affichage
      Pour I←1 à 48  Faire Ecrire('Wilaya ',I,'  Nombre  ',TW[I]) ; Fait ;
Fin.
```

**EXERCICE 1**

Soit le fichier NOMBRES.BIN qui contient une liste de nombres entiers. Écrire un algorithme qui affiche les nombres du fichier, leur somme et leur moyenne.

```
Algorithme Nombre;
Var F :Fichier de entier ;
    X,S,Nb :entier ; M :reel ;
Debut
   Assigner(F,'NOMBRES.BIN') ; Relire(F) ;
   Nb←0 ; S←0 ;
   Tantque Non FDF(F)
   Faire
        Lire(F,X) ; /*Lire un élément du fichier
        Ecrire(X) ; /*affichage à l'écran
        S←S+X ; Nb←Nb+1 ;
   Fait ;
   Si Nb≠0 Alors M←S/Nb ;
                    Ecrire('Somme des éléments =',S,'  Moyenne=',M)
            Sinon Ecrire('Fichier vide')
   Fsi ;
   Fermer(F) ;
Fin.
```

**EXERCICE 2**

1- Écrire un algorithme qui crée le fichier MOTS.TXT contenant une série de mots (longueur maximale d'un mot: 20 caractères). La saisie des mots se terminera à l'introduction du symbole **'*'** qui ne sera pas écrit dans le fichier.

2- Écrire un algorithme qui affiche le nombre de mots ainsi que la longueur moyenne des mots contenus dans le fichier MOTS.TXT.

3- Écrire un algorithme qui crée un deuxième fichier MOTS10.TXT contenant les mots du fichier MOTS.TXT de plus de 10 caractères.

```
Algorithme TraiteMot;
Var F,G :Fichier de chaine[20] ;
    X :chaine[20] ;
    Nb :entier ;     M :reel ;
Debut
   /*question 1
   Assigner(F,'MOTS.TXT') ; Reecrire(F) ; /*ouvrir F en écriture
   Ecrire('Donner une suite de mots. Introduire le mot '*' pour arrêter la saisie') ;
   Lire(X) ; /*Lire le premier mot à l'extérieur de la boucle
   Tantque X≠'*'
   Faire
        Ecrire(F,X) ;
        Lire(X) ; /*Lire le mot suivant
   Fait ;
   Fermer(F) ;
```

```
   /*question 2
   Nb←0 ; M←0 ; /*on peut utiliser M pour la somme des longueurs puis pour la moyenne
   Relire(F) /*ouvrire F en lecture
   Tantque Non FDF(F)
   Faire
        Lire(F,X) ; /*Lire un élément du fichier
        M←M+Taille(X) ; Nb←Nb+1 ;
   Fait ;

   Si Nb≠0      Alors   M←M/Nb ;
                        Ecrire('Nombre de  mots =',Nb,'  Longueur Moyenne=',M)
                Sinon   Ecrire('Fichier vide')
   Fsi ;
   Fermer(F) ;

   /*question 3
   Assigner(G,'MOTS10.TXT') ; Reecrire(G) ; Relire(F) /*ouvrir G en écriture et F en lecture
   Tantque Non FDF(F)
   Faire
        Lire(F,X) ; /*Lire un élément du fichier
        Si Taille(X)>10 Alors Ecrire(G,X) Fsi ;
   Fait ;
   Fermer(F) ; Fermer(G) ;
Fin.
```

**EXERCICE 3**

Considérons le type enregistrement suivant :

```
Type   Etudiant = Enregistrement
                        Matricule : entier ;
                        Nom, Prenom : chaine [20] ;
                        Moyenne : réel ;
                  Fin;
```

Soit **T** un tableau d'au plus 100 étudiants.

Ecrire un algorithme permettant de recopier tous les étudiants admis appartenant à **T** dans un fichier **ADMIS** de type étudiant. Un étudiant est admis si sa moyenne est supérieure ou égale 10.

```
Algorithme Etude ;
Type   Etudiant = Enregistrement
                        Matricule : entier ;
                        Nom, Prenom : chaine [20] ;
                        Moyenne : réel ;
      Fin;
Var     T :Tableau[1..100] de Etudiant ;
        F :Fichier de Etudiant ;
        X :Etudiant ;
        I,N :entier ;
```

```
Debut
   Ecrire('Donner le nombre d''etudiants') ;
   /*lecture des éléments du tableau
   Repeter Lire(N) Jusqu'à N>0 et N≤100 ;
   Pour I←1 à N
   Faire
        Avec X
        Faire   Lire(Matricule) ;
                Lire(Nom,Prenom) ;
                Lire(Moyenne) ;
        Fait ;
        T[I]←X ;
        /*On peut utiliser directement T[I] et on évite l'affectation ( en bleu )
        Avec T[I]
        Faire   Lire(Matricule) ;
                Lire(Nom,Prenom) ;
                Lire(Moyenne) ;
        Fait ;
   Fait ;
   /*création du fichier des admis
   Assigner(F,'ADMIS') ; Reecrire(F) ;
   Pour I←1 à N
   Faire
        Si T[I].Moyenne)≥10 Alors Ecrire(F,T[I]) Fsi;
        /*même remarque, on peut utiliser un enregistrement X
        X←T[I] ;
        Si X.Moyenne)≥10 Alors Ecrire(F,X)  Fsi ;
   Fait ;
   Fermer(F) ;
Fin.
```

**EXERCICE 4**

Soient les enregistrements suivants :

```
Type    TDate = Enregistrement  Jour, mois, année : entier ;  Fin;
       TDiscipline = Enregistrement  Discipline : chaine [10] ; Faculté : chaine [20] ; Fin;
        TEtudiant =  Enregistrement Nom, prenom : chaine [20] ;
                                    DateN : TDate ;
                                    Filiere : TDiscipline ;
                          Fin;
```

Soit **FEtudiant** un fichier d'étudiants. Ecrire un algorithme qui permet de :

- Remplir le fichier **FEtudiant.**
- Eclater le fichier **FEtudiant** en deux fichiers, **F1** (étudiants de la faculté '**FEI**') et **F2** (étudiants des autres facultés).

```
Algorithme Eclate ;
Type  TDate = Enregistrement  Jour, mois, année : entier ;  Fin;
      TDiscipline = Enregistrement  Discipline : chaine [10] ; Faculté : chaine [20] ; Fin;
```

```
    Etudiant =  Enregistrement      Nom, prenom : chaine [20] ;
                                    DateN : TDate ;
                                    Filiere : TDiscipline ;
                Fin;
Var   Etudiant :TEtudiant ;
      F,F1,F2 :Fuchier de TEtudiant ;
      FEI,Autre :Booléen ;
Debut
      Assigner(F,'FEtudiant') ; Réecrire(F) ;
      Avec Etudiant, Etudiant.DateN, Etudiant.Filiere
      Faire  Ecrire('Nom :') ;Lire(Nom) ; /*Lire 1er  nom à l'extérieur de la boucle
             Tantque Nom<>''
             Faire
                    Ecrire('Prénom :') ;Lire(prenom) ;
                    Ecrire('Date de naissance :') ;Lire(Jour,mois,Annee) ;
                    Ecrire('Discipline, Faculté :') ;Lire(Discipline, Faculté) ;
                    Ecrire(F,Etudiant) ;
                    Ecrire('Nom :') ;Lire(Nom) ; /*Lire le nom suivant
             Fait ;
      Fait ;
      Fermer(F) ;
      Relire(F) ;
      /*Pour éviter de créer des fichiers vides, on peut utiliser ces 2 indicateurs booléens, dans ce Cas
      /*l'assignation et la création ne se font que si nous trouvons un élément, après on remet le booléen à /*vrai
      pour  ne pas refaire ces opérations
      /*CETTE OPERATION N'EST PAS OBLIGATOIRE MAIS MIEUX LA SAVOIR
      FEI ←Faux ;   Autre ←Faux ;
      /*Si on n'utilise pas ces booléens, on doit assigner et ouvrir les 2 fichiers en écriture à ce niveau
      Assigner(F1,'FFEI') ; Réecrire(F1) ;
      Assigner(F2,'FAutrer'); Réecrire(F2);

      Si FDF(F) Alors  Ecrire('Fichier vide')
                Sinon  Tantque Non FDF(F)
                       Faire Lire(F,Etudiant) ;
                             Avec Etudiant.Filiere
                             Faire  Si Faculté ='FEI'
                                    Alors  Si Non FEI
                                           Alors Assigner(F1,'FFEI') ; Réecrire(F1) ; FEI ←Vrai Fsi ;
                                           Ecrire(F1,Etudiant) ;
                                    Sinon  Si Non Autre
                                           Alors Assigner(F2,'FAutrer'); Réecrire(F2); Autre←Vrai
                                           Fsi ;
                                           Ecrire(F2,Etudiant) ;
                                    Fsi ;
                             Fait ;
                       Fait ;
                       Fermer(F) ;
```

```
                    Si FEI Alors Fermer(F1)  Fsi ;
                    Si Autre Alors Fermer(F2) Fsi ;
        Fsi ;
Fin.
```

**EXERCICE 5**

1- Soient F1 et F2 deux fichiers d'entiers strictement positifs et sans répétition. Ecrire un algorithme qui construit (crée) un fichier G d'entiers tel que G contient pour chaque valeur de F1 la valeur et tous ses multiples appartenant à F2 (F1 et F2 sont supposés existants).

Exemple :

F1 : 3 10 20 17

F2 : 3 6 19 60 40 30

G : **3 3 6 60 30 10 60 40 30 20 60 40 17**

2- Ecrire un algorithme qui permet à partir du fichier résultat (G) de générer un autre fichier (H) contenant toutes les valeurs du fichier (G) (sans répétition) avec leur nombre.

Exemple :

H : 3 **2** 6 **1** 60 **3** 30 **2** 10 **1** 40 **2** 20 **1** 17 **1**

1-

```
Algorithme Multiple ;
Var      F1,F2,G : fichier de entier ;
         X,Y :entier ;
Debut
   Assigner(F1,'File1') ; Assigner(F2,'File2') ; Assigner(G,'File3') ;
   Relire(F1) ; Reecrire(G) ;
   Tantque Non FDF(F1)
   Faire
        Lire(F1,X) ; Ecrire(G,X) ;
        Relire(F2) ; /*revenir à chaque itération au début du fichier F2
        Tantque Non FDF(F2)
        Faire
            Lire(F2,Y) ;
            Si Y MOD X = 0 Alors Ecrire(G,Y) Fsi ;
        Fait ;
        Fermer(F2) ;
   Fait ;
   Fermer(F1) ; Fermer(G) ;
Fin.
```

2-

```
Algorithme ValRepetition ;
Var      G,H,G1,G2 : fichier de entier ;
         X,Y,N :entier :  Inter :booleen ;
```

```
Debut
   Assigner(G,'File3') ; Assigner(H,'FileH') ; Assigner(G1,'Inter1') ; Assigner(G2,'Inter2') ;
   Relire(G) ; Reecrire(G1) ;
   /*Copie de G dans G1, on utilise des fichiers intermédiaires
   Tantque Non FDF(G) Faire  Lire(G,X) ; Ecrire(G1,X) ; Fait ;
   Fermer(G) ; Fermer(G1) ;
   Relire(G1) ; Reecrire(H) ;
   Inter←Vrai ;
   Tantque Non FDF(G1)
   Faire
        Lire(G1,X) ; Ecrire(H,X) ;
        /* Traiter les répétitions et créer G2 (non traitées)
        Reecrire(G2) ; N ←1 ;
        Tantque Non FDF(G1)
        Faire
            Lire(G1,Y) ;
            Si X=Y  Alors  N ←N+1Sinon Ecrire(G2,Y) Fsi ;
        Fait ;
        /*Ecrire la répétition de X
        Ecrire(H,N) ;
        Fermer(G1) ; Fermer(G2) ;
        /*Copie de G2 dans G1 Pour traiter le reste. Ecraser l'ancien G1
        Relire(G2) ; Reecrire(G1) ;
        Tantque Non FDF(G2)  Faire  Lire(G2,X) ; Ecrire(G1,X) ; Fait ;

        /*ou bien pour optimiser, changer l'assignation des 2 fichiers G1 et G2 d'une manière alternée
        Inter← Non Inter ;
        Si Inter
        Alors   Assigner(G1,'Inter1') ; Assigner(G2,'Inter2') ;
        Sinon   Assigner(G1,'Inter2') ; Assigner(G2,'Inter1') ;
        Fsi ;

        Fermer(G1) ; Fermer(G2) ;
        /*Ré ouvrir G1 en lecture (non traitées)
        Relire(G1) ;
   Fait ;
   Fermer(G1) ; Fermer(H) ;
Fin.
```

**EXERCICE 6**

Soit F un fichier d'entiers représentant des séquences de nombres séparées par un ou plusieurs zéro. Ecrire un algorithme qui réalise les traitements suivants :
1- A partir de F (fichier existant), crée un fichier G contenant pour chaque séquence, la moyenne des nombres qui la constituent.
2- Puis, Supprimer les valeurs nulles du fichier G.
Exemple :

F : 0 0 **1 4 3 7** 0 0 0 **6 -9 2 7 -6** 0 **-10 3** 0 0

G : **3,75** **0,00** **-3,50** Avant suppression

G : **3,75** **-3,50** Après suppression

```
Algorithme TraiteFile ;
Var F :Fichier de entier;
    G,H :Fichier de reel ;
    X,S,Nb :entier ;  M :reel ;
Debut
    Assigner(F,'FileF') ; Relire(F) ;
    Assigner(G,'FileG') ;Reecrire(G) ;
    Tantque Non FDF(F)
    Faire
       Lire(F,X) ;
       Si X≠0
       Alors  S←0 ; Nb←0 ;
              Tantque Non FDF(F) et X≠0
              Faire  S←S+X ;
                     Nb←Nb+1 ;
                     Lire(F,X) ;
              Fait ;
              /*traitement du dernier élément non nul du fichier, la tête est sur FDF, mais X non traité ?
              Si X≠0 Alors S←S+X ; Nb←Nb+1 Fsi;
              M←S/Nb ;
              Ecrire(G,M) ;
       Fsi ;
    Fait ;
    Fermer(F) ; Fermer(G) ;
    /*Pour supprimer les valeurs nulles de G on utilise un fichier intermédiaire, ensuite on recopie ce
    /*fichier dans G
    Assigner(H,'Inter') ;Reecrire(H) ; Relire(G) ;
    Tantque Non FDF(G)
    Faire
       Lire(G,X) ;
       Si X≠0 Alors Ecrire(H,X) Fsi ;
    Fait ;
    Fermer(H) ; Fermer(G) ;

    /*copier H dans G
    Relire(H) ; Reecrire(G) ;
    Tantque Non FDF(H)
    Faire
       Lire(H,X) ;
       Ecrire(G,X) ;
    Fait ;
    Fermer(H) ; Fermer(G) ;
```

**Fin**.

**EXERCICE 7**

Soit F1 un fichier de caractères contenant une suite de télégrammes. Chaque télégramme est constitué d'une suite de mots séparés par un ou plusieurs blancs (^). Le télégramme se termine par le mot '**FINTEL**'.

Ecrire un algorithme qui permet, pour chaque télégramme, d'imprimer le texte, en respectant les conventions suivantes :
- Les mots imprimés seront séparés par un seul blanc.
- Les mots ne peuvent dépasser 12 caractères sinon ils seront tronqués à droite.
- Le texte de chaque télégramme est suivi de l'indication du nombre total de mots (tronqués ou non) et le nombre de mots tronqués.
- La fin de chaque télégramme sera indiquée par '**FINTEL**'.

**Version 1 : Affichage des télégrammes sur écran**

```
Algorithme Telegramme;
Var  F:fichier de caractère ;
     Nbmot, NbmotTq : entier;
     C:caractere;
     Mot:chaine;
     MotTq:chaine[12];
Debut
Assigner(F,'telegramme.txt') ; Relire(F) ;
Nbmot←0; NbmotTq←0;
Tantque Non FDF(F)
Faire
   Lire(F,C);
   Si C<>' '
   Alors
        Mot←'' /* initialiser Mot à vide
        Tantque (Non FDF(F) et C< >' ' )
        Faire
              Mot←Mot+C; /*utiliser la concaténation
              Lire(F,C);
        Fait;
        /*traiter le dernier caractère
        Si C< >' ' Alors Mot←Mot+C Fsi ;
        Si mot='FINTEL'
        Alors   /*traitement de la fin
              Ecrire(nbmot,' ',nbmotTq,' FINTEL') ;
              Si Non FDF(F) Alors Ecrire(' ')  Fsi ;
              Nbmot←0; NbmotTq←0;
        Sinon  /*traitement du mot
              Nbmot←Nbmot+1;
              Si Taille(Mot)>12
        Alors  MotTq←Mot ;  /* cette action permet de tronquer le mot
                    NbmotTq←NbmotTq+1;
```

```
                    Mot←MotTq
                Fsi ;
                Ecrire(Mot,' ')
        Fsi
    Fsi ;
Fait;
Fermer(F);
Fin.
```

**Version 2 : Création d'un fichier télégrammes en respectant les contraintes**

Dans ce **Cas** on aura besoin d'une **Fonction** pour convertir les entiers en chaine

```
Algorithme Telegramme;
Var  F,FC:Fichier de caractère ;
     Nbmot, NbmotTq,I,T : entier;
     C:caractere;
     Mot:chaine;
     MotTq:chaine[12];
Fonction IntToStr(X:entire):chaine;
Var R:entier; Ch:chaine;
Debut
        Ch←'';
        Repeter
                R←X MOD 10;
                X←X DIV 10;
                Cas R Vaut
        0:Ch←'0'+Ch;
        1:Ch←'1'+Ch;
        2:Ch←'2'+Ch;
        3:Ch←'3'+Ch;
        4:Ch←'4'+Ch;
        5:Ch←'5'+Ch;
        6:Ch←'6'+Ch;
        7:Ch←'7'+Ch;
        8:Ch←'8'+Ch;
        9:Ch←'9'+Ch;
        Fincas;
        Jusqu'à X=0;
        IntToStr←Ch;
Fin;

Debut
Assigner(F,'telegramme.txt') ; Relire(F) ;
Assigner(FC,'telegrammeC.txt') ; Reecrire(FC) ;
Nbmot←0; NbmotTq←0;
Tantque Non FDF(F)
Faire
```

```
    Lire(F,C);
    Si C<>' '
    Alors
        Mot←'' /* initialiser Mot à vide
        Tantque (Non FDF(F) et C< >' ' )
        Faire
                Mot←Mot+C; /*utiliser la concaténation
                Lire(F,C);
        Fait;
        /*traiter le dernier caractère
        Si C< >' ' Alors Mot←Mot+C Fsi ;
        Si Mot='FINTEL'
        Alors   /*traitement de la fin
                Mot←IntToStr(nbmot)+' '+IntToStr(nbmotTq)+' FINTEL' ;
                Si Non FDF(F) Alors Mot←Mot+' ' Fsi ;
        T←Taille(Mot) ;
                Pour I←1 à T Faire Ecrire(FC,Mot[I]) Fait ;
                Nbmot←0; NbmotTq←0;
        Sinon   /*traitement du mot
                Nbmot←Nbmot+1;
                Si Taille(Mot)>12
        Alors   MotTq←Mot ;  /* cette action permet de tronquer le mot
                        NbmotTq←NbmotTq+1;
                        Mot←MotTq
                Fsi ;
                Mot←Mot+' ' ; /* ajouter un blanc
                T←Taille(Mot) ;
                Pour I←1 à T Faire Ecrire(FC,Mot[I]) Fait
        Fsi
    Fsi ;
Fait;
Fermer(F); Fermer(FC) ;
Fin.
```

**EXERCICE 8**

Ecrire une procédure qui supprime le dernier élément du fichier F de nombre entiers.
En utilisant la procédure précédente, écrire un algorithme pour vider un fichier F existant de nombre entiers et de nom physique 'ESSAI.DAT', élément par élément. Cet algorithme devra afficher, après la suppression de chaque élément, la moyenne des éléments restants de F.

```
Type Fent :Fichier de entier ;

 /*l'assignation peut se faire in l'intérieur de la procédure, mais on préfère la faire à l'extérieur
Assigner(F,'FENT.DAT') ;

Procedure SupD(E/S/ F :Fent);
Var G :Fent ;    X,entier ;
```

```
Debut
    Relire(F) ;
    Si non FDF(F
    Alors   /*l'assignation du fichier intermédiaire se fait à l'intérieur, car c'est un fichier local
            Assigner(G,'G') ; Réécrire(G) ;
            Lire(F,x) ;
            /*  le fait de commencer par l'Ecriture avant la Lecture permet de copier les éléments de F
            /*dans G sauf  le dernier, on va le Lire, mais on ne l'écrit pas, la tête est sur FDF, si F
            /*contient un seul elt. Il ne sera pas copier dan G (la boucle ne s'exécute pas, on est sur
            /*FDF)
            Tantque non FDF(F)
            Faire
                    Ecrire(G,x) ;
                    Lire(F,x) ;
            Fait ;
            Fermer(F) ;   Fermer(G) ;
            /* Remplacer  le fichier F par G qui ne contient pas le dernier élément (copier G dans F)
            Relire(G) ;   Réécrire(F) ;
            Tantque non FDF(G)
            Faire
                    Lire (G,x) ;
                    Ecrire (F,x) ;
            Fait ;
            Fermer(G) ;
    Fsi ;
    Fermer(F) ;
Fin ;

Algorithme Phone ;
Var  F:fichier de entier ;
    X,S,Nb :entier ; M :reel ;
    Procedure SupD(E/S/ F :Fent);
    - - - - -
Debut
    Assigner(F,'ESSAI.DAT') ;
    Relire(F) ;
    Tant que non FDF(F)
    Faire
        SupD(F) ;
        Relire(F) ;
        S←0 ; Nb←0 ;
        Tant que non FDF(F)
        Faire
                Lire(F,X) ;
                S←S+X ; Nb←Nb+1 ;
        Fait ;
        Si Nb≠0 Alors M←S/Nb ;
```

```
                    Ecrire('Moyenne=',M)
                Sinon  Ecrire('Le fichier est vide')
            Fsi ;
        Fait ;
        Fermer(F) ;
Fin.
```

**EXERCICE 9**

Soient F1 et F2 deux fichiers de chaînes de caractères. Chaque chaîne représente un mot. Ecrire un algorithme qui construit un fichier F3, tel que F3 contient les mots de F1 qui n'existent pas dans F2.

```
Algorithme Mot123 ;
Var     F1,F2,F3 : fichier de chaine[30] ;
        X,Y :Chaine[30] ;
        Trouve :booleen ;
Debut
   Assigner(F1,'File1') ;  Assigner(F2,'File2') ; Assigner(F3,'File3') ;
   Relire(F1) ; Reecrire(F3) ;
   Tantque Non FDF(F1)
   Faire
       Lire(F1,X) ; /*Lire un mot de F1
       Trouve←Faux ; /*on suppose que le mot n'existe pas dans F2
       Relire(F2) ; /*revenir à chaque itération au debut du fichier F2
       Tantque Non FDF(F2) et Non Trouve
       Faire
          Lire(F2,Y) ;
          Si Y=X Alors Trouve ←Vrai Fsi ;
       Fait ;
       /* Si le mot n'est pas trouvé après le FDF de F2 on le met dans F3
       Si Non Trouve Alors Ecrire(F3,X) Fsi ;
       Fermer(F2) ;
   Fait ;
   Fermer(F1) ; Fermer(F3) ;
Fin.
```

**EXERCICE 10**

Soit le type suivant :

```
Type   Produit  =  Enregistrement
                       Code :  Entier ;
                       Désignation : Chaîne [ 80 ] ;
                       Prix : Réel ;
                   Fin ;
```

Soit F un fichier de produits. Ecrire une **Fonction** qui vérifie si les éléments de F sont triés par ordre croissant de leur Code.

```
Fonction Ftrie(F :fichier de Produit) :booleen ;
Var Eprod :Produit ;
    Code :entier ;
```

```
Debut
        Assigner(F,'Produit.dat') ; Relire(F) ;
        Ftrie ←Vrai ;
        Si Non FDF(F)
        Alors   Lire(F,Eprod) ;
                Tantque non FDF(F) et Ftrie
                Faire   code ←Eprod.code ;
                        Lire(F,Eprod) ;
                        Si code> Eprod.code  Alors Ftrie ← Faux Fsi ;
                Fait
        Fsi ;
Fin ;
```

**EXERCICE 11**

L'utilisation des téléphones portables permet de stocker le répertoire des contacts dans deux fichiers :
- Un fichier ' TEL.DAT ', enregistré sur la mémoire du téléphone ;
- Un fichier ' SIM.DAT ', enregistré sur la mémoire de la carte SIM.

Chaque fichier contient des références d'un contact regroupant : un nom, un prénom et un numéro de téléphone.
Les éléments des deux fichiers sont supposés déjà triés selon le numéro de téléphone.

1- Donnez la syntaxe (les instructions) d'assignation et d'ouverture des deux fichiers.

2- Ecrire une procédure qui permet de stocker les doublons dans un autre fichier. Un élément est un doublon s'il existe à la fois dans les deux fichiers.

1- Assigner(Ftel,'tel.dat') ; Relire(Ftel) ;
   Assigner(Fsim,'sim.dat') ; Relire(Ftel) ;

2-

```
Type    contact=Enregistrement
                        Nom,prenom :chaine[30] ;
                        Numero :chaine[10] ;
                Fin ;
Procedure Fdouble(E/Ftel,Fsim :fichier de contact ; S/ FD :fichier de contact) ;
Var     cont1,cont2 :contact ;
        Trouve,double :booleen ;
Debut
        Double ←vrai ;
        Tantque non FDF(ftel)
        Faire   Lire(Ftel,cont1) ;
                Relire(fsim) ; trouve ←faux ;
                Tantque non FDF(fsim) et non trouve
                Faire   Lire(fsim,cont2) ;
                        Si cont1=cont2  Alors   Si double
                                                Alors   Assigner(FD,'Fdouble.dat') ;
                                                        Reecrire(FD) ; double ←Faux
                                                Fsi ;
                                                Ecrire(FD,cont1) ;
                                                Trouve ←Vrai
                        Fsi ;
```

```
            Fait ;
            Fermer(fsim) ;
        Fait ;
        Fermer(Ftel) ;
        Si non double Alors fermer(FD) Fsi ;
Fin ;
```

**EXERCICE 12**

Soit **Fmot** un fichier de caractères alphabétiques contenant des mots séparés par un ou plusieurs caractères blanc.
1- Écrire une fonction ***Palindrome*** qui vérifie si un mot donné est un mot palindrome.
2- Écrire un algorithme qui affiche le nombre de mots palindromes et le plus court mot palindrome.

1-

```
Fonction Palindrome(mot :chaine) :booléen ;
Var     I,T :entier ;
Debut
        T←Taille(mot) ; Plaindrome ←vrai ; I ←1 ;
        Tantque  Palindrome et I< T DIV 2
        Faire   Si mot[I]<>mot[T-I+1]
        Alors Palindrome ←Faux Fsi ;
        I ←I+1 ;
        Fait ;
Fin ;
```

2-

```
Algorithme CourtPal ;
Type ch :chaine[50];
Var     Fmot :fichier de caractère ;
         tmin,t,NBpal :entier ;
        X :caractère ;
        Mot,cpal :ch ;
        Fonction Palindrome(mot :chaine) :booléen ;
        ------
Debut
        Assigner(Fmot,'fmot') ; relire(Fmot) ;
        tmin ←51 ;cpal ←'' ; NBpla← 0;
        Tantque non FDF(Fmot)
        Faire   Lire(Fmot,X) ;  /*récupérer un mot
                Mot←'' ; /*initialiser à vide
                Tantque non FDF(Fmot) et (X<>' '
                Faire
                   Mot←Mot+X ;
                   Lire(Fmot,X) ;
                Fait ;
                /*traiter le dernier caractère
                Si X<>' ' Alors Mot←Mot+X Fsi ;
```

```
            Si Mot<>''
            Alors
                  Si Palindrome(mot)
                  Alors  t ←Taille(mot) ; NBpal←NBpal+1 ;
                        Si t<tmin
                        Alors  tmin ←t ;
                              Cpal ←mot ;
                        Fsi ;
                  Fsi ;
            Fsi ;
      Fait ;
      Si NBcpal<>0
      Alors  Ecrire('Le nombre de mots palindrome est :',NBpal
             Ecrire('le plus court mot palindrome est :',cpal)
      Sinon  Ecrire('Pas de mots palindrome ou fichier est vide')
      Fsi ;
      Fermer(Fmot) ;
Fin.
```

**EXERCICE 13**

Soit F un fichier d'entiers (supposé existant) composé de séquences de nombres, chaque séquence est une répétition du même nombre (non nul). Toutes les séquences sont séparées par un zéro. Et aucune séquence du même nombre ne se répète dans le fichier.

| F | 14 | 14 | 14 | 0 | 5 | 5 | 0 | **29** | **29** | **29** | **29** | 0 | 6 | 6 | 6 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Position | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |

1- Ecrire une action paramétrée **Compresser** qui crée un fichier G d'enregistrement contenant pour chaque séquence le nombre représenté ainsi que la longueur de la séquence.

| G | Nombre | 14 | 5 | **29** | 6 |
|---|---|---|---|---|---|
| | Longueur | 3 | 2 | **4** | 3 |

2- En utilisant un seul parcours du fichier G et sans reparcourir le fichier F, trouver la position dans F de la plus longue séquence. (ex : Position = 8)

3- Ecrire une action paramétrée **Decompresser** qui permet de reconstruire un fichier H (de même type que F) à partir d'un fichier de même ne type que G.

1-

```
Type
Elcp=Enregistrement
        Val,Long :entier ;
      Fin ;
Fent : Fichier de Entier ;
Felcp : Fichier de Elcp ;
```

L'assignation se fait à l'extérieur comme suit :

```
Assigner(F,'Fname') ; Assigner(G,'Gname') ;
```

```
Procedure Compresser(E/S/ F : Fent ; E/S/ G : Felcp) ;
Var      X,I:Entier; EC:Elcp;
Debut
   Relire (F);  Réecrire (G);
   Tantque Non FDF(F)
   Faire          Lire (F,X) ;
          EC.Val ← X ; I ← 0 ;
          Tantque Non FDF(F) et X<>0
          FaireI  ← I+1 ;
       Lire (F,X) ;
          Fait ;
          Si FDF(F) Alors I ← I+1 Fsi;
          EC.Long ← I ; Ecrire (G,EC) ;
   Fait ;
   Fermer(F) ; Fermer(G) ;
Fin ;
```

2-

```
Fonction PosMax(G :Felcp) : Entier ;
Var   K,Max : Entier ;  EC :Elcp ;
Debut
   Relire(G) ; Max ← 0 ;   K ← 0 ;
   Tantque Non FDF(G)
   Faire   Lire(G,EC) ;
           Si EC.Long > Max
           Alors Max  ← EC.Long ;
                   PosMax  ← K+1
           Fsi ;
           K ← K+EC.Long+1;
   Fait ;
   Fermer(G) ;
Fin ;
```

3-

```
Procedure Decompresser(E/S/ G :Felcp ; E/S/ F : Fent) ;
Var    I:Entier; EC:Elcp;
Debut
   Relire(G) ;  Réecrire(F) ;
   Tantque Non FDF(G)
   Faire Lire(G,EC) ;
          Pour I  ← 1 à EC.Long
          Faire  Ecrire(F, EC.Val) ; Fait ;
          Si Non FDF(G) Alors Ecrire(F,0) Fsi ;
   Fait ;
   Fermer(G) ;  Fermer(F) ;
Fin ;
```

**EXERCICE 1**

Soit A une matrice (N,M) d'entiers. Ecrire un algorithme qui génère deux listes à partir de cette matrice.

1- La première regroupe les minimums des lignes (FIFO);
2- Et, la deuxième la somme des colonnes (LIFO).

```
Algorithme MatListe ;
Type   Pliste=^Liste ;
       Liste=Enregistrement
                Val :entier ;
                Suiv :Pliste ;
             Fin ;
Var    A :Tableau[1..100,1..150] de entier ;
       I,J,N,M,Min,S :entier ;
       Lmin,Lsom,P,Q :Pliste ;
Debut
       Répéter Lire(N) ; Jusqu'à (N>0) et (N≤100) ;
       Répéter Lire(M) ; Jusqu'à (M>0) et (M≤150) ;
       /*Lecture de la Matrice
       Pour  I ←1 à N  Faire  Pour  J ←1 à M  Faire
                                                Lire(A[I,J]) ;
                                              Fait ;
                       Fait ;
       /* Génération de la liste Lmin (FiFo)
       /*Créer la tête après le calcul du 1er min
       Min← A[1,1] ;
       Pour J←2 à M Faire Si A[1,J]<Min Alors Min← A[1,J] Fsi ; Fait ;
       Allouer(Lmin) ; Lmin^.Val←Min; P←Lmin ;
       Pour  I ←2 à N
       Faire  /*calcul du Min i
              Min← A[I,1] ;
              Pour J←2 à M Faire Si A[I,J]<Min Alors Min← A[I,J] Fsi ; Fait ;
              /*créer un élément de la liste et mettre à jour le chainage
              Allouer(Q) ;
              Q^.Val←Min ;  P^.Suiv←Q ; P←Q ;
       Fait ;
       P^.Suiv←Nil ;
       /* Génération de la liste Lsom (LiFo)
       Lsom←Nil ;
       Pour  J ←1 à M
       Faire  S←0 ;
              /* Calcul de la somme de colonne
              Pour I←1 à N Faire S←S+A[I,J] ; Fait ;
              /*créer un élément de la liste et mettre à jour le chainage
              Allouer(P) ;
              P^.Val←S ; P^.Suiv←Lsom ; Lsom←P ;
       Fait ;
Fin.
```

**EXERCICE 2**

Soit une liste d'entiers **L**, écrire les actions paramétrées suivantes permettant :

1- Le calcul du nombre d'éléments, et la détermination du maximum et du minimum ;
2- L'insertion d'une valeur val donnée dans une liste triée ;
3- La suppression des doublons (éléments identiques) ;
4- La création de la liste miroir de L (avec ensuite sans création d'une nouvelle liste) ;
5- La duplication d'une liste au début / à la fin;
6- La fusion de deux listes triées d'entiers L1 et L2 en une liste triée L3 ;

1-

```
Procedure CalListe(E/ L :Pliste ; S/ Nbl,Max,Min :entier) ;
Debut
        Nbl←0 ;
        Si L≠Nil
        Alors  Max←L^.Val ; Min←L^.Val ; Nbl←1 ; L←L^.Suiv ;
               Tantque L≠Nil
               Faire  Si L^.Val>Max
                      Alors Max←L^.Val
                      Sinon Si L^.Val<Min Alors Min←L^.Val Fsi
                      Fsi ;
                      Nbl←Nbl+1 ;
                      L←L^.Suiv ;
               Fait ;
        Fsi ;
Fin ;
```

2-

```
Procedure Insert(E/S/ L :Pliste ; E/ V :entier) ;
Var    P,Q,R:Pliste ;
Debut
        /*créer le nouveau élément à insérer
        Allouer(P) ; P^.Val←V ; P^.Suiv←Nil ;
        Si L=Nil
        Alors  L←P
        Sinon  Si V<L^.Val
               Alors  /*Insertion au début
                      P^.Suiv←L ;
                      L←P
               Sinon  /*chercher le lieu d'insertion, puis insérer, il faut garder le précédent (R)
                      R←L ; Q←L^.Suiv ;
                      Tantque Q≠Nil et V≥Q^.Val Faire R←Q ; Q←Q^.Suiv Fait ;
                      R^.Suiv←P ; P^.Suiv←Q
               Fsi
        Fsi ;
```

```
Fin ;
```

3-

```
Procedure SupDouble(E/L :Pliste) ;
Var P,Q,R :Pliste ;
Debut
        Si L≠Nil
        Alors
                P←L ;
                Tantque P^.Suiv≠Nil
                Faire  Q←P^.Suiv ; R←P ;
                        Tantque Q≠Nil
                        Faire  Si Q^.Val=P^.Val
                                Alors  R^.Suiv←Q^.Suiv ;
                                        Libérer(Q) ;
                                        Q← R^.Suiv
                                Sinon  R←Q ;
                                        Q←Q^.Suiv
                                Fsi ;
                        Fait ;
                        P←P^.Suiv ;
                Fait
        Fsi ;
Fin ;
```

4-

Création d'une nouvelle liste miroir (elle revient à créer une liste LIFO à partir de L)

```
Procedure MiroirNew(E/ L :Pliste ; S/ M :Pliste) ;
Var P :Pliste ;
Debut
        M←Nil ;
        Tantque L≠Nil
        Faire  Allouer(P) ;
                P^.Val←L^.Val ;
                P^.Suiv←M ;
                M←P ;
                L←L^.Suiv ;
        Fait ;
Fin ;
```

Créer une liste miroir de L sans nouvelles allocation (ça revient à inverser la liste).

```
Procedure Inverser(E/S/ L :Pliste) ;
Var P,Q :Pliste ;
Debut
        Si L≠Nil
        Alors  Q←L^.Suiv ; L^.Suiv←Nil ;
                Tantque Q≠Nil
                Faire  P←Q^.Suiv;  Q^.Suiv←L ;
                        L←Q ;  Q←P ;
                Fait ;
```

```
        Fsi ;
Fin ;
```

5-
On peut utiliser un paramètre DF pour le type de duplication (Debut :0 ; Fin :1) pour ne pas écrire 2 Pro.

```
Procedure DupliqueD(E/S/ L :Pliste ; E/ DF :entier) ;
Var P,Q,R,T,T0 :Pliste ;
 Debut
        Si L≠Nil
        Alors  T←L ; /*sauvegarder la tête L
               /*Créer une liste FIFO à partir de L
               Allouer(P) ; P^.Val←T^.Val ; Q←P ; /*P est la tête de la nouvelle liste
               T0←T ; T←T^.Suiv ;
               Tantque T≠Nil
               Faire  Allouer(R) ; R^.Val←T^.Val ;
                      Q^.Suiv←R ;
                      Q←R ;
                      T0←T ; /*T0 est le dernier élément de L
                      T←T^.Suiv ;
               Fait ;
               Si DF=0
               Alors  /*Relier la fin de la liste créée au début de la liste initiale (L)
                      Q^.Suiv←L ;
                      L←P  /*changer la tête de la liste initiale
               Sinon  /*Relier dernier élément de L (T0) au début de la liste
                      Q^.Suiv←Nil ;
                      T0^.Suiv←P
               Fsi ;
        Fsi ;
Fin ;
```

6-
Fusion de 2 listes triées dans une nouvelle liste

```
Procedure FusionNew(E/ L1,L2 :Pliste ; S/ L :Pliste) ;
Var P,Q :Pliste ;
Debut
        Si L1≠Nil et L2≠Nil
        Alors  /*créer 1er élément de L
               Allouer(L) ;
               Si L1^.Val<L2^.Val  Alors  L^.Val←L1^.Val ; L1←L1^.Suiv
                                   Sinon  L^.Val←L2^.Val ; L2←L2^.Suiv
               Fsi ;
               P←L ;
               /*parcourir les 2 listes
               Tanque L1≠Nil et L2≠Nil
               Faire  Allouer(Q) ;
                      Si L1^.Val<L2^.Val  Alors  Q^.Val←L1^.Val ; L1←L1^.Suiv
                                          Sinon  Q^.Val←L2^.Val ; L2←L2^.Suiv
                      Fsi ;
                      P^.Suiv←Q ;
```

```
                P←Q ;
        Fait ;
        /*continuer avec L1 ou avec L2
        Tantque L1≠Nil
        Faire   Allouer(Q) ;
                Q^.Val←L1^.Val ; L1←L1^.Suiv ; P^.Suiv←Q ;
                P←Q ;
        Fait ;
        Tantque L2≠Nil
        Faire   Allouer(Q) ;
                Q^.Val←L2^.Val ; L2←L2^.Suiv ; P^.Suiv←Q ;
                P←Q ;
        Fait ;
        P^.Suiv←Nil
    Sinon  /*une des 2 liste est vide ou les 2 sont vides
        L←Nil ;
        Si L1≠Nil
        Alors  /*créer 1er élément de L
                Allouer(L) ;
                L^.Val←L1^.Val ; L1←L1^.Suiv ; P←L ;
                Tantque L1≠Nil
                Faire   Allouer(Q) ;
                        Q^.Val←L1^.Val ; L1←L1^.Suiv ; P^.Suiv←Q ;
                        P←Q ;
                Fait ;
                P^.Suiv←Nil ;
        Fsi ;

        Si L2≠Nil
        Alors  /*créer 1er élément de L
                Allouer(L) ;
                L^.Val←L2^.Val ; L2←L2^.Suiv ;   P←L ;
                Tantque L2≠Nil
                Faire   Allouer(Q) ;
                        Q^.Val←L2^.Val ; L2←L2^.Suiv ;  P^.Suiv←Q ;
                        P←Q ;
                Fait ;
                P^.Suiv←Nil ;
        Fsi ;
    Fsi ;
Fin ;
```

Fusion de deux listes triées dans une 3eme liste sans allocation

```
Procedure FusionNew(E/ L1,L2 :Pliste ; S/ L :Pliste) ;
Var P :Pliste ;
Debut
    Si L1≠Nil et L2≠Nil
    Alors  /*créer 1er élément de L
        Si L1^.Val<L2^.Val  Alors  L←L1 ;
                                   L1←L1^.Suiv
```

```
                                    Sinon  L←L2 ;
                                           L2←L2^.Suiv
            Fsi ;
            P←L ;
            /*parcourir les 2 listes
            Tanque L1≠Nil et L2≠Nil
            Faire  Si L1^.Val<L2^.Val  Alors  P^.Suiv ←L1; P←L1 ;
                                              L1←L1^.Suiv
                                       Sinon  P^.Suiv ←L2; P←L2 ;
                                              L2←L2^.Suiv
                   Fsi ;
            Fait ;
            /*continuer avec L1 ou avec L2
            Si L1≠Nil      Alors P^.Suiv ←L1
                           Sinon P^.Suiv ←L2
            Fsi
      Sinon /*une des 2 liste est vide ou les 2 sont vides
            Si L1≠Nil      Alors L←L1
                           Sinon L←L2
            Fsi ;
Fin ;
```

**EXERCICE 3**

Soit T un tableau de 26 listes de chaînes de caractères. La liste 1 contient des mots commençant par la lettre 'A', la liste 2 contient des mots commençant par la lettre 'B'…etc.
Déclarer T et écrire une **Fonction** permettant de vérifier l'existence d'un mot M dans la structure.

```
Fonction Trouve(E/ T :Tliste ; E/ M :Chaine[25]) :Booleen ;
Var    L :Cliste ;
Debut
       /*récupérer la tête de liste des mots commençant avec le même caractère que M
       L←T[M[1]] ;
       /*Recherche de M
       Trouve←Faux ;
       Tantque L≠Nil et Non Trouve
       Faire  Si L^.Mot=M Alors Trouve←Vrai Fsi ;
              L←L^.Suiv ;
       Fait ;
Fin ;
```

**EXERCICE 4**

Soit L une liste d'entiers positifs. Ecrire une procédure qui permet d'éclater la liste L en deux listes : Lp contenant les entiers pairs et Li contenant les entiers impairs. (Sans création de nouvelles listes).

```
Procedure Eclate(E/ L :Pliste ; S/ Lp,Li :Pliste) ;
Var Pp,Pi :Pliste ;
Debut
       Lp←Nil ; Li←Nil ;
       Tantque L≠Nil
       Faire  Si L^.Val MOD 2=0    Alors  /*vérifier si la tête Lp est créée
                                          Si Lp=Nil   Alors      Lp←L ;
```

```
                                        Pp←L
                                Sinon   Pp^.Suiv←L ;
                                        Pp←L
                        Fsi
                Sinon   /*vérifier si la tête Li est créée
                        Si Li=Nil   Alors   Li←L ;
                                            Pi←L
                                    Sinon   Pi^.Suiv←L ;
                                            Pi←L
                        Fsi
            Fsi ;
            L←L^.Suiv ;
        Fait ;
        Si Lp≠Nil Alors Pp^.Suiv←Nil Fsi ;
        Si Li≠Nil  Alors Pi^.Suiv←Nil Fsi ;
Fin ;
```

**EXERCICE 5**

Soient deux listes d'entiers L1 et L2 :

1- Ecrire une fonction qui vérifie si L1 et L2 sont identiques (contiennent les mêmes éléments dans le même ordre).
2- Ecrire une fonction qui vérifie si L1 est incluse dans L2 (tous les éléments de L1 se trouvent dans L2, ici l'ordre ne compte pas).
3- Ecrire une fonction qui vérifie si L1 et L2 sont disjointe (L1 ∩ L2 = Ø).

1)

```
Fonction Idem(L1,L2 :Pliste) :Booleen ;
Debut
        Idem←Vrai ;
        Tantque L1≠Nil et L2≠Nil et Idem
        Faire   Si L1^.Val≠L2^.Val Alors Idem←Faux Fsi ;
                L1←L1^.Suiv ;
                L2←L2^.Suiv ;
        Fait ;
        Si L1≠Nil ou L2≠Nil Alors Idem←Faux Fsi ;
Fin ;
```

2)

```
Fonction Incluse(L1,L2 :Pliste) :Booleen ;
Var     T :Pliste ;
Debut
        Incluse ←Vrai ;
        Tantque L1≠Nil et Incluse
        Faire   Incluse←Faux ;
                T←L2 ; /*pour revenir à chaque au début de la liste L2
                Tantque L2≠Nil Non Incluse
                Faire   Si L1^.Val=T^.Val Alors Incluse←Vrai Fsi ;
                        T←T^.Suiv ;
                Fait ;
                L1←L1^.Suiv ;
        Fait ;
Fin ;
```

3)

```
Fonction Disjoint(L1,L2 :Pliste) :Booleen ;
Var     T :Pliste ;
Debut
        Disjoint ←Vrai ;
        Tantque L1≠Nil et Disjoint
        Faire   T←L2 ; /*pour revenir à chaque au début de la liste L2
                Tantque T≠Nil et Disjoint
                Faire   Si L1^.Val=T^.Val Alors Disjoint ←Faux Fsi ;
                        T←T^.Suiv ;
                Fait ;
                L1←L1^.Suiv ;
        Fait ;
Fin ;
```

**EXERCICE 6**

Soient deux listes L1 et L2 de valeurs entières positives :
Ecrire une action paramétrée permettant de déplacer (sans allocation ni libération) les valeurs paires de L1 vers L2, et de déplacer les valeurs impaires de L2 vers L1 ;

```
Procedure Deplacer(E/S/ L1,L2 :Pliste) ;
Var P,Q,T2:Pliste ;
Debut
        /*détacher les valeurs paires de L1 et les mettre au début de L2
        Q←L1 ;P←L1 ; T2←L2 ; /*sauvegarder la tête L2
        Tantque Q≠Nil
        Faire   Si Q^.Val MOD 2=0
                Alors   Si Q=L1
                        Alors   Q^.Suiv←L2 ; L2←Q ;
                                Q←Q^.Suiv ;
                                L1←Q
                        Sinon   P^.Suiv←Q^.Suiv ;
                                Q^.Suiv←L2 ; L2←Q ;
                                Q←P^.Suiv
                        Fsi
                Sinon   P←Q ;
                        Q←Q^.Suiv
                Fsi ;
        Fait ;
        /*à ce niveau L1 est vide ou ne contient que des éléments impairs
        /*détacher les valeurs impaires de L2 et Les mettre au début de L1
        /*chercher la dernière valeur paire insérée dans L2 si elle existe
        Si T2=L2
        Alors   Q←L2 ; P←L2
        Sinon   Q←T2 ; P←L2
                Tantque P^.Suiv≠T2 Faire Q←Q^.Suiv ; Fait
        Fsi ;
        /*Optimisation : Traitement de L2 à partir de T2
        Tantque Q≠Nil
        Faire   Si Q^.Val MOD 2≠0
                Alors   Si Q=P
                        Alors   Q^.Suiv←L1 ; L1←Q ;
                                Q←Q^.Suiv ;
                                L2←Q
                        Sinon   P^.Suiv←Q^.Suiv ;
```

```
                        Q^.Suiv←L1 ; L1←Q ;
                        Q←P^.Suiv
                Fsi
        Sinon   P←Q ;
                Q←Q^.Suiv
        Fsi ;
    Fait ;
Fin ;
```

**EXERCICE 7**

Soit ***NOTES*** un fichier de notes contenant le résultat du module algorithmique (fichier existant), tel que l'ième élément du fichier contient la note obtenue par l'étudiant numéro ***i***.

1- Écrire une action paramétrée ***CREATION*** qui construit, à partir du fichier ***NOTES***, une liste linéaire chainée **L** contenant tous les étudiants (pour chaque étudiant on garde le numéro et la note obtenue).

2- Ecrire une action paramétrée ***SUPPRESSION*** permettant la suppression les étudiants ajournés de la liste L (ne garder que les étudiants admis).

```
Type    Pnote=^NoteAlgo ;
        NoteAlgo=Enregistrement
                        Num :entier ;
                        Note :reel ;
                        Suiv :Pnote ;
                    Fin ;
        Fnote :Fichier de reel ;
```

1)

On considère que l'assignation du fichier est à l'extérieur : Assigner(F,'Notes') ;

```
Procedure Creation(E/ F :Fnote ; S/ L :Pnote) ;
Var     P,Q :Pnote ;
        X :reel ; I :entier ;
Debut
        L←Nil ;
        Relire(F) ;
        Si non FDF(F)
        Alors   /*créer la tête de liste (fifo)
                Lire(F,X) ; I←1 ;
                Allouer(L) ;
                L^.Num←I ; L^.Note←X ; P←L ;
                /*créer les autres éléments de la liste
                Tantque Non FDF(F)
                Faire   Lire(F,X) ; I←I+1 ;
                        Allouer(Q) ;
                        Q^.Num←I ; Q^.Note←X ;
                        P^.Suiv←Q ;
                        P←Q ;
                Fait ;
                P^.Suiv←Nil
        Fsi ;
        Fermer(F) ;
Fin ;
```

2)

```
Procédure Suppression(E/S/ L :Pnote) ;
Var     P,Q :Pnote ;
```

```
Debut
        Si L≠Nil
        Alors   /*suppression début
                Tantque L≠Nil et L^.Note<10
                Faire   P←L ; L←L^.Suiv ; Libérer(P) ; Fait ;
                /*suppression ailleurs, on doit garder trace du précédent (P)
                P←L ; Q←L ;
                Si P≠Nil Alors Q←P^.Suiv Fsi ;
                Tantque Q≠Nil
                Faire   Si Q^.Note<10 Alors   P^.Suiv←Q^.Suiv ;
                                              Q←P^.Suiv
                                      Sinon   P←Q ;
                                              Q←Q^.Suiv
                        Fsi ;
                Fait
        Fsi ;
Fin ;
```

**EXERCICE 8**

Soit L une liste d'étudiants. Chaque étudiant est défini par son matricule (entier) son nom et prénom (chaine) et sa filière ('ACAD','GTR' ou 'ISIL').

1- Donner la déclaration de cette liste.

2- Ecrire une procédure **ECLATE** qui permet d'éclater la liste L en deux listes L1 contenant les étudiants de la filière 'GTR' et L2 contenant les étudiants de la filière 'ISIL'. A la fin, la liste originale L contiendra les étudiants de la filière 'ACAD' seulement.

1-

```
Type    PEtud=^Etudiant ;
        Etudiant=Enregistrement
                        Matricule :entier ;
                        Nom,Prenom :chaine[25] ;
                        Filiere :chaine[4] ;
                        Suiv :PEtud ;
                    Fin ;
```

L est du **Type** PEtud

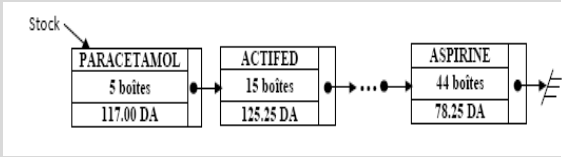
2-

```
Procedure Eclate(E/S/ L,L1,L2 :PEtud) ;
Var     P1,P2,LnP,Lp :PEtud ;
Debut
        L1←Nil ; L2←Nil ; Ln←Nil ;
        Tantque L≠Nil
        Faire   Si L^.Filiere='GTR'
                Alors   Si L1=Nil
                        Alors   L1←L ; P1←L
                        Sinon   P1^.Suiv←L ; P1←L
                        Fsi
                Sinon   Si L^.Filiere='ISIL'
                        Alors   Si L2=Nil
                                Alors   L2←L ; P2←L
                                Sinon   P2^.Suiv←L ; P2←L
                                Fsi
                        Sinon   Si Ln=Nil
```

```
                        Alors   Ln←L ; Lp←L
                        Sinon   Lp^.Suiv←L ; Lp←L
                        Fsi
                Fsi ;
                L←L^.Suiv ;
        Fait ;
        L←Ln ;
        Si Ln≠Nil Alors Lp^.Suiv←Nil Fsi ;
        Si L1≠Nil Alors P1^.Suiv←Nil Fsi ;
        Si L2≠Nil Alors P2^.Suiv←Nil Fsi ;
Fin ;
```

**EXERCICE 9**

Un pharmacien souhaite traiter les informations concernant son stock de médicaments par ordinateur. On vous propose de représenter ces informations sous forme de liste linéaire chainée où chaque élément contient le libellé d'un médicament, la quantité disponible (nombre de boites) et le prix unitaire.




1- Donner les structures de données nécessaires a la représentation de ce stock (voir schéma).
2- Ecrire la procédure **Vendre** (Med, NbBoites) permettant de retirer, **Si** possible, 'NbBoites' du médicament 'Med' du stock. (Il faut supprimer du stock le médicament dont la quantité atteint 0).
3- Ecrire la procédure **Acheter** (Med, NbBoites, Prix) permettant au pharmacien d'alimenter son stock par 'NbBoites' du médicament 'Med' ayant le prix unitaire 'Prix' DA. On considère qu'un médicament prenne toujours le nouveau prix. **Si** le médicament n'existe pas, il faut l'insérer.
4- Ecrire la fonction **ValStock** permettant de calculer la valeur des médicaments dans le stock.

1-

```
Type    Pmedic=^Medic ;
        Medic=Enregistrement
                Libele :chaine[15] ;
                Qte :entier ;
                Prix :reel ;
                Suiv :Pmedic ;
              Fin ;
```

2-

```
Procedure Vendre(E/S/ M :Pmedic ;E/ Med :chaine[15] ; E/ NB :entier ; S/ Possible :booleen) ;
Var     P :Pmedic ;
Debut
        Possible←Faux ;
        Si M≠Nil
        Alors   P←M ; Q←M ;
                /*recherche de Med
                Tantque Q≠Nil et Q^.Libele≠Med Faire P←Q ; Q←Q^.Suiv ; Fait ;
                Si Q≠Nil
                Alors   /*on a trouvé Med, on teste si la vente est possible
                        Si Q^.Qte≥NB
                        Alors   Possible←Vrai ;
                                Q^.Qte←Q^.Qte-NB ;
                                /*si qte devient nulle, on supprime
                                Si Q^.Qte=0
                                Alors   Si P=Q Alors   M←P^.Suiv ; Libérer(P)  /*Cas sup Tête
```

```
                                        Sinon   P^.Suiv←Q^.Suiv ; Libérer(Q)
                                    Fsi
                                Fsi
                            Fsi
                        Fsi
        Fsi ;
Fin ;
```

3-

```
Procedure Acheter(E/S/ M :Pmedic ; E/ Med :chaine[15] ; E/ NB :entier ; E/ Prix :reel) ;
Var P,Q :Pmedic ;
Debut
        Si M=Nil
        Alors   Allouer(M) ;
                M^.Libele←Med ;
                M^.Qte←NB ;
                M^.Prix←Prix ;
                M^.Suiv←Nil
        Sinon   /*Recherche de Med
                P←M ; Q←M ;
                /*recherche de Med
                Tantque Q≠Nil et Q^.Libele≠Med Faire P←Q ; Q←Q^.Suiv ; Fait ;
                Si Q≠Nil
                Alors   /*on a trouvé Med, on met à jour les données
                        Q^.Qte←Q^.Qte+NB ;
                        Q^.Prix←Prix
                Sinon   /*Med n'existe pas, on le crée à la fin, après le dernier élément (P)
                        Allouer(Q) ;
                        Q^.Libele←Med ; Q^.Qte←NB ; Q^.Prix←Prix ;
                        Q^.Suiv←Nil ;
                        P^.Suiv←Q
                Fsi ;
        Fsi ;
Fin ;
```

4-

```
Fonction ValStock(M :Pmedic) :reel ;
Debut
        ValStock←0 ;
        Tantque M≠Nil
        Faire   ValStock←ValStock+M^.Qte*M^.Prix ; M←M^.Suiv ; Fait ;
Fin ;
```

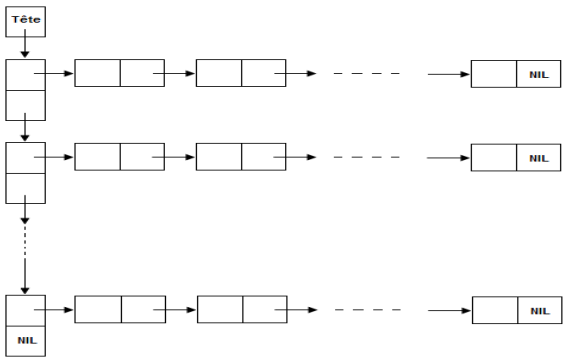
**EXERCICE 10**

Soit la structure ci-dessous représentant une liste de listes d'entiers.

1- Ecrire une fonction qui vérifie Si deux listes d'entiers L1 et L2 sont identiques ;

2- Ecrire une action paramétrée qui supprime une liste d'entiers de la structure ;

3- Ecrire un Algorithme qui supprime toutes les listes identiques en double (Chaque liste d'entiers de la structure doit être unique).



Déclaration de la structure (PlisteV)

```
Type
Pliste=^ListeH ;
ListeH=Enregistrement
        Val :entier ;
        Suiv :Pliste ;
        Fin ;

PlisteV=^ListeV ;
ListeV=Enregistrement
        SuivH :Pliste ;
        SuivV :PlisteV ;
        Fin ;
```

1-

```
Fonction Idem(L1,L2 :Pliste) :Booleen ;
Debut
        Idem←Vrai ;
        Tantque L1≠Nil et L2≠Nil et Idem
        Faire   Si L1^.Val≠L2^.Val Alors Idem←Faux Fsi ;
                L1←L1^.Suiv ;
                L2←L2^.Suiv ;
        Fait ;
        Si L1≠Nil ou L2≠Nil Alors Idem←Faux Fsi ;
Fin ;
```

2-

```
Procedure SupprimeH(E/S/ LV :PlisteV ; E/S/ LH :Pliste) ;
Var     T,TC:PlisteV ;
        P :Pliste ;
Debut
   Si LV≠Nil
   Alors
        Si LV^.SuivH=LH
        Alors   /*supprimer la tête après la suppression de la liste H
                Tantque LH≠Nil Faire P←LH ; LH←LH^.Suiv ; Libérer(P) ; Fait ;
                T←LV ; LV←LV^.SuivV ; Libérer(T)
        Sinon   /*chercher LV qui contient LH
                T←LV ; TC← T^.Suiv ;
                Tantque TC≠Nil
                Faire   Si TC^.SuivH=LH
                        Alors   Tantque LH≠Nil Faire P←LH ; LH←LH^.Suiv ; Libérer(P) ; Fait ;
                                T^.SuivV←TC^.SuivV ; Libérer(TC)
                        Sinon   T←TC ; TC← TC^.Suiv
                        Fsi ;
                Fait
        Fsi ;
   Fsi ;
Fin ;
```

3-

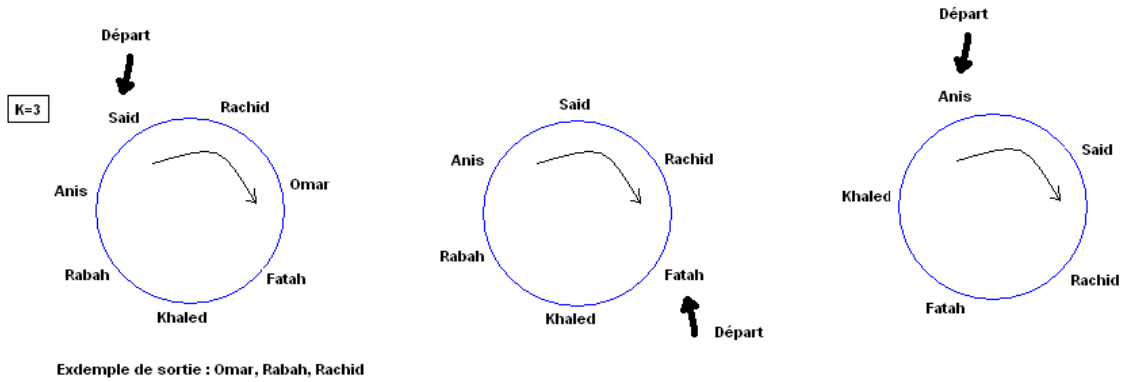
```
Procedure SuppDoubleH(E/S/ LV :PlisteV) ;
Var     L1,L2 :Pliste ;
        T,TC,TR :PlisteV ;
```

```
Debut
        T←LV ;
        Si T≠Nil
        Alors   L1←T^.SuivH ;
                TR←T ;
                TC←T^.SuivV ;  /*TR est le précédent de TC
                Tantque TC≠Nil
                Faire   L2←TC^.SuivH ;
                        Si Idem(L1,L2)
                        Alors   /*avant de libérer TC dans la procédure on sauvegarde son Suiv pour la suite
                                TC←TC^.SuivV ;
                                SupprimeH(LV,L2)
                        Sinon   TR←TC ; TC←TC^.SuivV
                        Fsi ;
                Fait ;
                T←T^.SuivV
        Fsi ;
Fin ;
```

**EXERCICE 11**

Pour jouer à Am-Stram-Gram, *n* enfants forment une ronde, ils choisissent l'un d'entre eux comme le premier, commencent à compter à partir de lui et décident que le $k^{ième}$ doit quitter la ronde, puis à nouveau le $k^{ième}$ en partant du suivant, et ainsi de suite.

Ecrire une fonction qui donne la suite (sous forme d'une liste chaînée simple) des enfants dans l'ordre où ils sont sortis de la ronde.




Exdemple de sortie : Omar, Rabah, Rachid

La ronde des enfants est représentée par une liste chaînée de chaînes. (Il peut être intéressant de modifier cette liste pour la rendre circulaire !).

```
Pronde=^Ronde
Ronde=Enregistrement
        Nom :chaine[20] ;
        Suiv :Pronde ;
        Fin ;

Fonction Sortie(L :Pronde ; K :entier ) :Pronde ;
Var     I :entier ;
        P,Q, :Pronde ;
Debut
```

```
        Sortie←Nil ;
        Si L≠Nil
        Alors  /*transformer L en liste circulaire
               P←L ;
               Tantque P^.Suiv≠Nil Faire P←P^.Suiv ; Fait ; P^.Suiv←L ;
               /*Création de la liste de sortie en mode FiFo
               Sortie←Nil ;
               Si L≠Nil
               Alors  Tantque L^.Suiv≠L
                      Faire  Pour I←1 à K-2 Faire L←L^.Suiv ; Fait ;
                             P←L^.Suiv ;  /* élément sortant
                             L^.Suiv ←P^.Suiv ;  /*détacher P
                             Si Sortie=Nil
                             Alors  Sortie←P ; Q←P  /*créer la tête, Q : dernier élément
                             Sinon  Q^.Suiv←P ; Q←P  /*créer les autres
                             Fsi ;
                             L← L^.Suiv ;
                      Fait ;
                      Si Sortie=Nil   Alors Sortie←L        /*la liste contient 1 seul élément
                                      Sinon Q^.Suiv←L
                      Fsi ;
                      L←Nil
               Fsi ;
        Fsi ;
Fin ;
```

**EXERCICE 12**

Soit L une liste d'entiers.

1- Ecrire une procédure **DETACHE** qui renvoie l'adresse du maximum de la liste L et le détache de la liste sans le supprimer.

2- En utilisant la procédure **DETACHE**, écrire une procédure **TRIER** qui trie la liste L dans un ordre décroissant (sans création de nouvelle liste).

1-

```
Procedure Detache(E/S/ L :Pliste ; S/ Pmax:Pliste) ;
Var     Max :entier ;
        Prx,P,Q :Pliste ;
Debut
        Pmax←L ;
        Si L≠Nil
        Alors  P←L ; Max←P^.Val ; Pmax←L ;
               Q←P^.Suiv ;
               Tantque Q≠Nil
               Faire  Si Q^.Val>Max Alors Max← Q^.Val ; Pmax←Q ;Prx←P  Fsi ;
                      P←Q ; Q←Q^.Suiv ;
               Fait ;
               Si Pmax=L
               Alors  L←L^.Suiv
               Sinon  Prx^.Suiv←Pmax^.Suiv
```

```
                Fsi ;
        Fsi ;
Fin ;
```

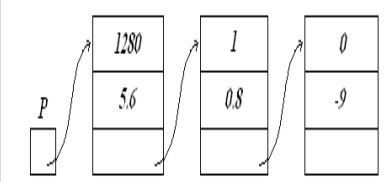
2-

```
Procedure Trier(E/S/ L :Pliste) ;
Var     T,P,Q :Pliste ;
Debut
        /*Tri dans l'ordre décroissant en utilisant DETACHE  implique la création d'une liste FIFO
        T←Nil ;
        Si L≠Nil
        Alors   /*Créer la tête
                Detache(L,P) ;
                T←P ;
                /*créer les autres éléments
                Tantque L≠Nil
                Faire   Detache(L,Q) ;
                        P^.Suiv←Q ;
                        P←Q ;
                Fait ;
                P^.Suiv←Nil ;
                L←T
        Fsi ;
Fin ;
```

**EXERCICE 13**

Dans cet exercice, on se propose de développer un module permettant de manipuler des polynômes creux. Un polynôme creux est un polynôme contenant très peu de monômes non nuls.
Exemple : $P(x) = 5.6\ x^{1280} + 0.8\ x - 9$ contient 1281 termes dont 3 seulement sont non nuls.
Chaque monôme est décrit par un enregistrement de type *Tmonome* comportant les 3 champs suivants :
- *Deg* : entier représentant le degré du monôme ;
- *Coef* : réel représentant le coefficient du monôme ;
- *Suiv* : pointeur sur le monôme suivant.

La liste représentant le polynôme sera triée par ordre de degré décroissant. Une liste vide (Nil) correspond au polynôme zéro ;




1- Ecrire une procédure **Ajouter** qui ajoute à un polynôme (Pol) la valeur d'un monôme défini par son degré (Deg) et son coefficient (Coef).
2- Ecrire les procédures **Somme** et **Produit** qui réalisent respectivement la somme et le produit de deux polynômes Pol1 et Pol2.
3- Ecrire une **Fonction Valeur** qui calcule la valeur du polynôme pour une valeur val de la variable x.
4- Ecrire une procédure **Derive** qui détermine la dérivée DPol d'un polynôme Pol.
5- Ecrire une procédure **Primitive** qui détermine la primitive PPol d'un polynôme Pol, sachant que PPol(0)=1.

**Type**

```
Tmonome=^Monome ;
Monome=Enregistrement
                Coef :reel ;
                Deg :entier ;
```

```
            Suiv :Tmonome ;
        Fin ;
1-
Procedure Ajouter(E/S/ Pol :Tmonome ; E/ M :Tmonome) ;
Var   P,Q :Tmonome ;
Debut
  Si M^.Coef≠0
  Alors
      Si Pol=Nil
      Alors  Pol←M ; M^.Suiv←Nil
      Sinon  /*recherche de position d'ajout ou mettre à jour le Coef dans les =
             Si M^.Deg>Pol.Deg
             Alors  M^.Suiv←Pol ; Pol←M  /*ajouter au début
             Sinon  Si Pol^.Deg=M^.Deg
                    Alors  /*mettre à jour le Coef, puis libérer M
                           Pol^.Coef←Pol^.Coef+M^.Coef ;
                           Libérer(M)  /*en Cas d'égalité, on n'a plus besoin de M
                           /*Si le Coef devient nul, on supprime le monôme
                           Si Pol^.Coef=0 Alors P←Pol ; Pol←Pol^.Suiv ; Liberer(P) Fsi
                    Sinon  P←Pol ; Q←P^.Suiv ;
                           Tantque Q≠Nil et Q^.Deg>M^.Deg
                           Faire P←Q ; Q←Q^.Suiv ; Fait ;
                           Si Q=Nil
                           Alors  P^.Suiv←M ; M^.Suiv←Nil
                           Sinon  Si Q^.Deg=M^.Deg
                                  Alors  Q^.Coef←Q^.Coef+M^.Coef ;
                                         Libérer(M) /*en Cas d'égalité, on n'a plus besoin de M
                                         /*Si le Coef devient nul, on supprime le monôme
                                         Si Q^.Coef=0
                                         Alors  P^.Suiv←Q^.Suiv ; Liberer(Q) Fsi
                                  Sinon  P^.Suiv←M ; M^.Suiv←Q
                           Fsi
                    Fsi
             Fsi
      Fsi
  Fsi ;
Fin ;

2-
Procedure Somme(E/ Pol1,Pol2 :Tmonome ; S/ Pol3 :Tmonome) ;
Var   M :Tmonome ;
Debut
      /*Ajouter tous les monômes de Pol1 à Pol3== Duplication de Pol1 Dans Pol3
      Pol3←Nil ;
      Tantque Pol1≠Nil
      Faire  Allouer(M) ; M^.Deg←Pol1^.Deg ; M^.Coef←Pol1^.Coef ;
             Ajouter(Pol3,M) ;
             Pol1←Pol1^.Suiv ;
      Fait ;
      /*Ajouter tous les monômes de Pol2à Pol3
      Tantque Pol2≠Nil
```

```
        Faire  Allouer(M) ; M^.Deg←Pol2^.Deg ; M^.Coef←Pol2^.Coef ;
               Ajouter(Pol3,M) ;
               Pol2←Pol2^.Suiv ;
        Fait ;
Fin ;

Procedure Produit(E/ Pol1,Pol2 :Tmonome ; S/ Pol3 :Tmonome) ;
Var     M,P :Tmonome ;
Debut
   Pol3←Nil ;
   Si Pol≠Nil et Pol2≠Nil
   Alors
        /*Multiplier chaque monôme de Pol2 par Pol1 et ajouter à Pol3
        Tantque Pol2≠Nil
        Faire  P←Pol1 ;
               Tantque P≠Nil
               Faire  Allouer(M) ;
                      M^.Deg← Pol2^.Deg +P^.Deg ; M^.Coef← Pol2^.Coef *P^.Coef ;
                      Ajouter(Pol3,M) ;
                      P←P^.Suiv ;
               Fait ;
               Pol2←Pol2^.Suiv ;
        Fait ;
Fin ;

3-
Fonction Valeur(E/ Pol :Tmonome ; E/ X :reel ):reel ;
Var     I : entier ;
        Px : reel ;
Debut
        Valeur←0 ;
        Tantque Pol≠Nil
        Faire  Px←1 ;
               Pour I←1 à Pol^.Deg Faire Px←Px*X ; Fait ; /*calcul de puissance de X
               Valeur←Valeur+Pol^.Coef*Px ;
               Pol←Pol^.Suiv ;
        Fait ;
Fin ;

4-
Procedure Derive(E/ Pol :Tmonome ; S/ DPol :Tmonome) ;
Var     M,P :Tmonome ;
Debut
        Dpol←Nil ;
        Si Pol≠Nil
        Alors  /*créer la le premier monôme de DPol
               Si Pol^.Deg≠0
               Alors  Allouer(Dpol) ;
                      DPol^.Coef←Pol^.Coef*Pol^.Deg ;
                      DPol^.Deg←Pol^.Deg-1 ;
                      P←Dpol
```

```
            Fsi ;
            Pol←Pol^.Suiv ;
            /*créer les autres monômes de DPol
            Tantque Pol≠Nil
            Faire  Si Pol^.Deg≠0
                   Alors  Allouer(M)) ;
                          M^.Coef←Pol^.Coef*Pol^.Deg ;
                          M^.Deg←Pol^.Deg-1 ;
                          P^.Suiv←M ;
                          P←M ;
                   Fsi ;
                   Pol←Pol^.Suiv ;
            Fait ;
            Si Dpol≠Nil Alors P^.Suiv←Nil Fsi
      Fsi ;
Fin ;
```

- On peut aussi calculer la Dérivé en utilisant **Ajouter,** mais la première solution est plus **optimale**.

```
Procedure Derive(E/ Pol :Tmonome ; S/ DPol :Tmonome) ;
Var   M,P :Tmonome ;
Debut
      Dpol←Nil ;
      Tantque Pol≠Nil
      Faire  Si Pol^.Deg≠0
             Alors  Allouer(M)) ;
                    M^.Coef←Pol^.Coef*Pol^.Deg ;
                    M^.Deg←Pol^.Deg-1 ;
                    Ajouter(DPol,M) ;
             Fsi ;
             Pol←Pol^.Suiv ;
      Fait ;
Fin ;
```

5-

```
Procedure Primitive(E/ Pol :Tmonome ; S/ PPol :Tmonome) ;
Var   M,P :Tmonome ;
Debut
      Ppol←Nil ;
      Si Pol≠Nil
      Alors  /*créer la le premier monôme de PPol
             Allouer(Ppol) ;
             PPol^.Coef←Pol^.Coef/(Pol^.Deg+1) ;
             PPol^.Deg←Pol^.Deg+1 ;
             P←Ppol ;
             Pol←Pol^.Suiv ;
             /*créer les autres monômes de PPol
             Tantque Pol≠Nil
             Faire  Allouer(M)) ;
                    M^.Coef← Pol^.Coef/(Pol^.Deg+1) ;
                    M^.Deg←Pol^.Deg+1 ;
```

```
                P^.Suiv←M ;
                P←M ;
                Pol←Pol^.Suiv ;
            Fait ;
        Fsi ;
        /*créer la constante
        Allouer(M) ; M^.Deg←0 ; M^.Coef←1 ; M^.Suiv←Nil ; /*PPol(0)=1
        Si Ppol≠Nil Alors P^.Suiv←M Sinon PPol←M Fsi ;
Fin ;
```

- On peut aussi calculer la primitive en utilisant **Ajouter,** mais la première solution est plus **optimale**.

```
Procedure Primitive(E/ Pol :Tmonome ; S/ PPol :Tmonome) ;
Var     M,P :Tmonome ;
Debut
        Ppol←Nil ;
        Tantque Pol≠Nil
        Faire  Allouer(M)) ;
               M^.Coef← Pol^.Coef/(Pol^.Deg+1) ;
               M^.Deg←Pol^.Deg+1 ;
               Ajouter(Ppol,M) ;
               Pol←Pol^.Suiv ;
        Fait ;
        /*créer la constante
        Allouer(M) ; M^.Deg←0 ; M^.Coef←1 ; M^.Suiv←Nil ; /*PPol(0)=1
        Si Ppol≠Nil Alors P^.Suiv←M Sinon PPol←M Fsi ;
Fin ;
```

**EXERCICE 14**

Définition : Une matrice est qualifiée de creuse (Anglais : sparse) si elle comporte majoritairement des coefficients (éléments) nuls.

Exemple : matrice creuse

$$\begin{pmatrix} \mathbf{1} & \mathbf{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} \\ 0 & \mathbf{3} & \mathbf{9} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & \mathbf{4} & 0 & 0 & 0 \end{pmatrix}$$

On peut représenter une matrice creuse en ne tenant compte que des éléments non nuls en utilisant une liste chainée. En utilisant la représentation d'une matrice creuse par une liste :

1- Donner la déclaration de la structure de données nécessaire.

2- Ecrire la procédure permettant de Remplir une telle structure pour une matrice creuse A(M,N) donnée.

3- Ecrire l'action paramétrée permettant de calculer la somme de deux matrices ainsi représentées.

4- Ecrire une procédure permettant d'afficher une matrice ainsi représentée.

1-

```
Type    PMat=^Matrice ;
        Matrice=enregistrement
                I,J,Val :entier ;
```

```
        Suiv :pliste ;
    Fin ;
```

2-

```
Procedure RemplirA(E/S/ A:Pmat) ;
Var   P,Q :pliste ;
      I,J,V :entier ;
Début
      A←Nil ;
      Lire(I,J,V) ;
      Si V<>0
      Alors  /*créer la tête
             Allouer(A) ;
             A^.I ←I ; A^.J ←J ; A^.Val ←V ;
             P←A;
             /*créer le reste des éléments, on considère que les valeurs sont introduite ligne par ligne
             Lire(I,J,V) ;
             Tant que V<>0
             Faire  Allouer(Q) ;
                    Q^.I ←I ; Q^.J ←J ; Q ^.Val ←V ;
                    P^.Suiv←Q;
                    P ←Q ;
                    Lire(I,J,V) ;
             Fait ;
             P^.Suiv←Nil
      Fsi ;
Fin ;
```

3-

```
Procedure Somme(E/ A,B :Pmat ; S/ C :Pmat) ;
Var   P,Q,R,E :Pmat ;
Debut
      /*Ca création de la matrice C revient à faire une fusion des 2 listes A et B en se basant sur les indices
      C←Nil ;
      Tantque A≠Nil et B≠Nil
      Faire  Si A^.I<B^.I
             Alors  Allouer(E) ;
                    E^.I←A^.I ;E^.J←A^.J ; E^.Val←A^.Val ;
                    A←A^.Suiv
             Sinon  Si A^.I>B^.I
                    Alors  Allouer(E) ;
                           E^.I←B^.I ;E^.J←B^.J ; E^.Val←B^.Val ;
                           B←B^.Suiv
                    Sinon  /*A^.I=B^.I, on passe à J
                           Si A^.J<B^.J
                           Alors  Allouer(E) ;
                                  E^.I←A^.I ;E^.J←A^.J ; E^.Val←A^.Val ;
                                  A←A^.Suiv
```

```
                        Sinon  Si A^.J>B^.J
                               Alors  Allouer(E) ;
                                      E^.I←B^.I ;E^.J←B^.J ; E^.Val←B^.Val ;
                                      B←B^.Suiv
                               Sinon  /*A^.I=B^.I et A^.J=B^.J, on vérifie si la somme≠0
                                      Si (A^.Val+B^.Val)
                                      Alors  Allouer(E) ;
                                             E^.I←B^.I ;E^.J←B^.J ; E^.Val←A^.Val+B^.Val ;
                                      Fsi
                        Fsi
                    Fsi
                Fsi ;
                Si C=Nil
                Alors  /*premier élément
                       C←E ; P←C
                Sinon  /*autre élément
                       P^.Suiv←E ; P←E
                Fsi ;
        Fait ;
        /*continuer avec A ou avec B
        Tantque A≠Nil
        Faire  Allouer(E) ;
                E^.I←A^.I ;E^.J←A^.J ;
                E^.Val←A^.Val ;
                A←A^.Suiv ;
                Si C=Nil
                Alors  /*premier élément
                       C←E ; P←C
                Sinon  /*autre élément
                       P^.Suiv←E ; P←E
                Fsi ;
        Fait ;
        Tantque B≠Nil
        Faire  Allouer(E) ;
                E^.I←B^.I ;E^.J←B^.J ;
                E^.Val←B^.Val ;
                B←B^.Suiv ;
                Si C=Nil
                Alors  /*premier élément
                       C←E ; P←C
                Sinon  /*autre élément
                       P^.Suiv←E ; P←E
                Fsi ;
        Fait ;
        Si C≠Nil Alors P^.Suiv←Nil Fsi ;
Fin ;
```

4-

```
Procedure AfficheA(E/ A :Pmat ) ;
Debut
        Tantque A≠Nil
```

```
        Faire   Ecrire('A[',P^.I,',', P^.J,']=', P^.Val) ;
                P ←P^.Suiv ;
        Fait ;
Fin ;
```

**EXERCICE 15**

Soit L une liste de caractères constituant des mots (un mot est une suite de caractères ne contenant pas de blanc) séparés par un seul caractère blanc (espace).
Ecrire une procédure qui inverse les mots de la liste L sans création d'une nouvelle liste.

```
Type
Cliste=^Lcar ;
Lcar=Enregistrement
        car :caractère ;
        Suiv :Cliste ;
        Fin ;
Procedure InverseM(E/S/ L :Cliste) ;
Var     D,F,P,Q,B :Cliste ;
Debut
        D←L ;
        Si L≠Nil
        Alors  /*traitement premier mot
                F←L^.Suiv ;
                Tantque F≠Nil et F^.Car≠' '
                Faire  P←F^.Suiv;
                       F^.Suiv←L ;
                       L←F ;
                       F←P ;
                Fait ;
                D^.Suiv←F ;
                /* traitement des autres mots
                Q←F ;
                Tantque Q≠Nil
                Faire  B←Q ; /*si Q≠Nil il est sur un blanc B
                       Q←Q^.Suiv ; D←Q ;  /*D est le premier caractère du mot
                       F←Q^.Suiv ;
                       Tantque F≠Nil et F^.Car≠' '
                       Faire  P←F^.Suiv;
                              F^.Suiv←Q ;
                              Q←F ;
                              F←P ;
                       Fait ;
                       B^.Suiv←Q ;  /*Q est sur le dernier caractère du mot
                       Q←F ; /*F  est sur un blanc ou Nil
                       D^.Suiv←Q ;
                Fait ;
        Fsi ;
Fin ;
```