Correction TD 4 - les Tableaux -
ASD1 / LGLSI1 - LIRIS1

Ex 1 :

```
procédure Insertion(A : Tab, B : Tab, n : Entier, var n1 : Entier)
   Var
      R : réel
      i : Entier
   Début
      Ecrire('donner un réel R :')
      lire(R)
      i ← 1
      tantque (i <= n et A[i] < R) faire
            B[i] ← A[i]
            i ← i+1
      fin tq
      B[i] ← R
      tantque (i <= n) faire
            B[i+1] ← A[i]
            i ← i+1
      fin tq
      n1 ← n+1
fin
```

Ex2 :

```
procédure Eclater (T: tab, TP: Tab, TN: Tab, n: Entier, Var np: Entier, nn: Entier)
  Var
    i : Entier
  Début
    np ← 0
    nn ← 0
    pour i de 1 à n faire
      Si (T[i] < 0) Alors
        nn ← nn+1
        TN[nn] ← T[i]
      Sinon
        np ← np+1
        TP[NP] ← T[i]
      finsi
    fin pour
  fin
```

Ex3 :

1)
```
fonction Saisie_Taille () : Entier
  Var
    Dim : Entier
  Début
    Répéter
      Ecrire ("donner une dimension")
      lire (Dim)
    Jusqu'à (Dim > 1 et Dim <= 50)
    Saisie_Taille ← Dim
  fin
```

2) procédure Saisie_Tab(T: tab, Dim: Entier)

```
Var
  i : Entier
Début
  pour i de 1 à Dim faire
      Ecrire ("donner un entier :")
      lire (T[i])
  finpour
fin
```

3) procédure Permuter (Var X, Y : Entier)

```
Var
  z : Entier
Début
  z ← x
  x ← y
  y ← z
fin
```

procédure Tri (T: tab, Dim : Entier)

```
Var
  i, j : Entier
Début
  pour i de 1 à N-1 faire
      pour j de i+1 à N faire
          si (T[i] > T[j]) Alors
              Permuter (T[i], T[j])
          finsi
      fin pour
  finpour
fin
```

4/ procédure Fusion (T1:tab, T2:tab, F: Tab, N: Entier, M:Entier)

```
Var
  i, j, k : Entier
Début
    i ← 1
    j ← 1
    k ← 1
    tantque (i<=N et j<=M) f
        Si (T1[i] <= T2[j]) Alors
            F[k] ← T1[i]
            i ← i+1
            k ← k+1
        Sinon
            F[k] ← T2[j]
            j ← j+1
            k ← k+1
        finsi
    fintq
    tantque (i<=N) f
        F[k] ← T1[i]
        i ← i+1
        k ← k+1
    fintq
    tantque (j<=M) f
        F[k] ← T2[j]
        j ← j+1
        k ← k+1
    fintq
fin
```

5) procédure Affiche (T : Tab, Dim : Entier)

```
Var
   i : Entier
Début
   pour i de 1 à Dim faire
      Ecrire (T[i])
   finpour
fin
```

6) Algorithme Ex3

```
type
   Tab : tableau [1..1w] d'entier
Var
   A, B, C : Tab
   N, M : Entier
Debut
   N ← SaisirTaille()
   M ← SaisirTaille()
   SaisirTab(A, N)
   SaisirTab(B, M)
   Tri(A, N)
   Tri(B, M)
   Fusion(A, B, C, N, M)
   Affiche(C, N+M)
fin
```

EX4

1) procédure changement (T: tab, N: Entier)

```
Var
  i: Entier
Début
  pour i de 1 à N faire
    Répéter
      Ecrire("donner le prix d'un produit:")
      lire(T[i])
    Jusqu'à (T[i] > 0)
  finpour
fin
```

2) fonction somme (T: tab, N: Entier): Réel

```
Var
  i: Entier
  S: réel
Début
  S ← 0
  pour i de 1 à N faire
    S ← S + T[i]
  fin pour
  somme ← S
fin
```

3) fonction Mini (X: réel, Y: réel): réel

```
Début
  si (X <= Y) Alors
    Mini ← X
  Sinon
    Mini ← Y
  finsi
fin
```

4) fonction Ideal (T1: tab, T2: tab, N: Entier): réel

```
Var
    i: Entier
    S: réel
Début
    S ← 0
    pour i de 1 à N faire
        S ← S + Mini (T1[i], T2[i])
    fin pour
    Ideal ← S
fin
```

5) procédure Affiche (T1: tab, T2: Tab, N: Entier)

```
Début
    Ecrire ("Prix du panier d'après le 1er Super:", Somme(T1, N))
    Ecrire ("Prix panier selon 2ème super:", Somme(T2, N))
    Ecrire ("Prix ideal du panier:", Ideal (T1, T2, N))
fin
```

6) Algorithme EX4

```
type tab: tableau [1..20] de réel
Var
    T1, T2: tab
    N: Entier
Début
    Répéter
        Ecrire ("donner le nb de produits:")
        Lire (N)
    Jusqu'à (N > 2 et N < 20)
    Changement (T1, N)
    Changement (T2, N)
    Affiche (T1, T2, N)
fin
```

Ex5 :

1) procédure Saisie_Tab(T : tab, N : Entier)
```
Var
  i : Entier
Début
  pour i de 1 à N faire
    Ecrire ("donner un élément :")
    Lire (T[i])
  fin pour
fin
```

2) procédure Afficher(T : tab, N : Entier)
```
Var
  i : Entier
Début
  pour i de 1 à N faire
    Ecrire (T[i])
  fin pour
fin
```

3) procédure Recherche (T : tab, N : Entier)
```
Var
  i, pos, nb : Entier
Début
  pos ← 1
  nb ← 1
  pour i de 2 à N faire
    si (T[i] < T[pos]) Alors
      pos ← i
      nb ← 1
    sinon
      si (T[i] = T[pos]) Alors
        nb ← nb + 1
      finsi
    finsi
  fin pour
  Ecrire ("nb d'occ =", nb, "Ind 1ère occ =", pos)
fin
```

4) fonction Déterminer (T: tab, N: Entier) : Entier

```
Var
  nb, i, j : Entier
Début
  nb ← 0
  pour i de 1 à N faire
      j ← i+1
      tantque (j <= N et T[i] <> T[j]) faire
          j ← j+1
      fintq
      si (j > N) Alors
          nb ← nb+1
      finsi
  finpour
  Déterminer ← nb
fin
```

5) procédure Eliminer (T: tab, Var N: Entier)

```
Var
  i, j, k : Entier
Début
  i ← 1
  tantque (i <= N) faire
      j ← i+1
      tantque (j <= N) faire
          si (T[j] = T[i]) Alors
              pour k de j à N-1 faire
                  T[k] ← T[k+1]
              finpour
              N ← N-1
          sinon
              j ← j+1
      fintq
      i ← i+1
  fintq
fin
```

6) Algorithme Ex5

```
type
   Tab: tableau [1..100] d'entier
Var
   T : Tab
   N : Entier
Début
   Répéter
      Ecrire ("donner le nb d'elt:")
      Lire (N)
   Jusqu'à (N>0 et N<=100)
   Saisie_tab(T,N)
   Recherche(T,N)
   Si ( Déterminer(T,N) <> N) Alors
         Eliminer(T,N)
   finsi
   Afficher (T,N)
fin
```

Ex 6 :

1) procédure Saisir_TAB (T : Tab, N : Entier)

```
Var
  i : Entier
Début
  pour i de 1 à N faire
    Répéter
      Ecrire ("donner un élt :")
      Lire (T[i])
    Jusqu'à (T[i] > 0)
  fin pour
fin
```

2) procédure Afficher (T : Tab, N : Entier)

```
Var
  i : Entier
Début
  pour i de 1 à N faire
    Ecrire (T[i])
  fin pour
fin
```

) fonction Tester (T : Tab, N : Entier) : Entier

```
Var
  i, Res : Entier

Début
  Res ← 3
  i ← 2
  tantque (i <= N et T[i] = T[i-1]) faire
    i ← i+1
  fintq
  si (i <= N)
    si (T[i] > T[i-1]) Alors
      Res ← 1
      tantque (i <= N et T[i] >= T[i-1]) faire
        i ← i+1
      fintq
```

```
        sinon
            Res ← 2
            tantque (i <= N et T[i] <= T[i-1]) faire
                i ← i+1
            fintq
        finsi
        si (i <= N) Alors
            Res ← 4
        finsi
    finsi
    TestN ← Res
fin.
```

4/
```
fonction Determine(T: Tab; N: Entier, X: Entier): Entier
  Var
    nb, i: Entier
  Début
    nb ← 0
    pour i de 1 à N faire
        si (T[i] = X) Alors
            nb ← nb+1
        finsi
    fin pour
    Déterminé ← nb
fin
```

5)
```
procédure Trier(T: Tab; N: Entier)
  Var
    X, i, F: Entier
    permute: booléen
  Début
  F ← N-1
```

```
Répéter
    permute ← faux
    pour i de 1 à F faire
        Si (T[i] > T[i+1]) Alors
            x ← T[i]
            T[i] ← T[i+1]
            T[i+1] ← x
            permute ← Vrai
        finsi
    fin pour
    F ← F-1
Jusqu'à (permute = faux)
```

6)
```
procédure Compresser (T: tab, TC: tab, N: Entier, Var NC: Entier)
var
    i, j, nb: Entier

Début
    i ← 1
    j ← 1
    Trier (T, N)
    tantque (i <= N) faire
        nb ← Déterminer (T, N, T[i])
        TC[j] ← T[i]
        TC[j+1] ← nb
        i ← i + nb
        j ← j + 2.
    fintq.
    NC ← j - 1
fin
```

7/ Algorithme Ex6

```
Type
    Tab1: tableau [1..1w] d'entier
    Tab2: tableau [1..2w] d'entier

Var
    T : tab1
    TC : tab2
    N, NC : Entier

Début
    Répéter
        Ecrire("donner la taille de T:")
        lire(N)
    Jusqu'à (N>0 et N<=1w)
    Saisie_Tab(T, N)
    selon (Tester(T, N)) faire
        1 : Ecrire("tableau trié de l'ordre croissant")
        2 : Ecrire("tableau trié de l'ordre décroissant")
        3 : Ecrire("tableau constant")
        4 : Ecrire("tableau quelconque")
    fin selon
    Compresser(T, TC, N, NC)
    Afficher(TC, NC)

fin
```

Ex 7

```
Algorithme Ex7
  Const
    longmax = 5

  type
    Matrice : tableau [1..longmax][1..longmax] d'entier

  Var
    M : Matrice
    nb, S, P, i, j, ind : Entier

  Début
    Répéter
      Ecrire ("donner nb : ")
      lire (nb)
    Jusqu'à (nb > 0 et nb <= longmax)
    pour i de 1 à nb faire
      pour j de 1 à nb faire
        Ecrire ("donner un élt : ")
        lire (M[i][j])
      fin pour
    fin pour
    Répéter
      Ecrire ("donner ind : ")
      lire (ind)
    Jusqu'à (ind >= 1 et ind <= nb)
    S ← 0
    pour i de 1 à nb faire
      S ← S + M[ind][i]
    fin pour
    P ← 1
    pour i de 1 à nb faire
      P ← P * M[i][ind]
    fin pour
    Ecrire (S)
    Ecrire (P)
fin
```

Ex 8 :

```
procédure Somme (M1: Matrice, M2: Matrice, M3: Matrice, n: Entier,
                 m: Entier)
  Var
    i, j : Entier
  Début
    pour i de 1 à n faire
      pour j de 1 à m faire
        M3[i][j] ← M1[i][j] + M2[i][j]
      fin pour
    fin pour
  fin
```

Ex 9 :

1)
```
procédure Saisi_NB (Var NB: Entier)
  Début
    Répéter
      Ecrire ("donner un nombre: ")
      lire (NB)
    Jusqu'à (NB > 0 et NB <= 10)
  fin
```

2)
```
procédure Chargement (A: Matrice, N: Entier, M: Entier)
  Var
    i, j : Entier
  Début
    pour i de 1 à N faire
      pour j de 1 à M faire
        Ecrire ("donner un élt: ")
        lire (A[i][j])
      fin pour
    fin pour
  fin
```

3) procédure Multiplication (A: Matrice, B: Matrice, C: Matrice, N: Entier, M: Entier, P: Entier)

```
Var
  i, j, k, S: Entier
Début
  pour i de 1 à N faire
    pour j de 1 à P faire
      S ← 0
      pour k de 1 à M faire
        S ← S + A[i][k] * B[k][j]
      fin pour
      C[i][j] ← S
    fin pour
  fin pour
fin
```

) procédure Affichage (A: Matrice, N: Entier, M: Entier)

```
Var
  i, j : Entier
Début
  pour i de 1 à N faire
    pour j de 1 à M faire
      Ecrire (A[i][j])
    fin pour
  fin pour
fin
```

5) Algorithme Ex9

```
type
    matrice : tableau [1..10][1..10] d'entier
Var
    A, B, C : Matrice
    N, M, P : Entier
    Saisie_NB(N)
    Saisie_NB(M)
    Saisie_NB(P)
    Chargement(A, N, M)
    Chargement(B, M, P)
    Multiplication(A, B, C, N, M, P)
    Affichage(A, N, M)
    Affichage(B, M, P)
    Affichage(C, N, P)
```

fin

Ex 10

Algorithme Symétrique

```
type
    Matrice : tableau [1..10][1..10] d'entier
var
    M, i, j : Entier
    Sym1, Sym2 : booléen
    A : Matrice

Début
    Répéter
        Ecrire ("donner la dimension de la matrice:")
        Lire (M)
    Jusqu'à (M > 0 et M <= 10)
    pour i de 1 à M faire
        pour j de 1 à M faire
            Lire (A[i][j])
        fin pour
    fin pour
    Sym1 ← vrai
    i ← 1
    tantque (i < M et Sym1 = vrai) faire
        j ← i + 1
        tantque (j <= M Et Sym1 = vrai) faire
            Si (A[i][j] <> A[j][i]) Alors
                Sym1 ← faux
            Sinon
                j ← j + 1
            finsi
        fintq
        i ← i + 1
    fintq.
```

```
Sym2 ← Vrai
i ← 1
tantque (i < M Et Sym2 = Vrai) faire
    j ← i
    tantque (j <= M-i Et Sym2 = Vrai) faire
        si (A[i][j] <> A[M-j+1][M-i+1]) Alors
            Sym2 ← faux
        finsi
        j ← j+1
    fintq
    i ← i+1
fintq
Si (Sym1 = Vrai) Alors
    Ecrire ("la matrice est symétrique par
            rapport à la 1ère diagonale")
finsi
Si (Sym2 = Vrai) Alors
    Ecrire ("la matrice est symétrique
            par rapport à la 2ème diagonale")
finsi
fin
```