Exercice 1 :

Soit un type TEMPS contenant le résultat obtenu par un athlète dans une course de marathon.

1- Ecrire une fonction TRANSFORM qui transforme un temps T de type TEMPS en un entier S exprimant ce temps en secondes.

2- Ecrire une AP DECOMPOS qui décompose un temps S exprimé en secondes en un temps T de type TEMPS.

3- Soient N participants dans cette course (N≤100). Chaque athlète est défini par un Numéro, un Nom et un Résultat de type TEMPS. Ecrire un algorithme permettant de construire un vecteur contenant les informations des différents athlètes, puis affiche la liste des athlètes (avec résultat) sélectionnés ayant obtenu un résultat inférieur ou égale à la moyenne de tous les athlètes.

```
Type
       Temps = Enregistrement
          H, M, S :entier ;
       Fin;

Fonction SommeT(T1,T2 :Temps) : Temps
Var
      X : Entier ;
      T : Temps;
Debut
      X <--- T1.S+T2.S;
      T.S <--- X mod 60;
      X <--- X div 60;
      X <--- X+T1.M+T2.M;
      T.M <--- X mod 60;
      T.H <--- X div 60+ T1.H+T2.H;

      Retourner T;
Fin ;

Fonction Transform(T :Temps) : Entier
Debut
     Retourner (T.S + 60*T.M + 3600*T.H) ;
Fin ;

Fonction Decompos(S : Entie) :Temps
Var
     T : Temps;
  Debut
     T.H <--- S div 3600 ;
     S <--- S mod 3600 ;
     T.M <--- S div 60 ;
     T.S <--- S mod 60 ;

     Retourner T;
Fin;

5-
Algorithme CalculT;
Type
     Temps=Enregistrement
        H,M,S :entier ;
     Fin ;
    Athlete = Enregistrement
```

```
        Num : Entier;
        Nom : Chaine[50];
        Res : Temps;
    Fin;
    TabAthlete = Tableau[1..100] de Athlete;
Var
     TAth : TabAthlete ;
     i,N, moy : Entier ;

Fonction SommeT(T1,T2 :TEMPS) : TEMPS
       ...
Fin;

Fonction Transform(T :TEMPS) : Entier
       ...
Fin;

Fonction Decompos(S : Entier) :TEMPS
       ...
Fin;

Debut
     Repeter
          Ecrire("Donner N Nombre d'athletes compris entre 1 et 100");
          Lire(N);
     Jusqu'a ((N >= 1) et (N <= 100));
     Pour i <-- 1 a N  Faire
         Avec  T[i], T[i].Res  Faire
              Ecrire('Donner Numero, Nom, et Temps T : H M S du ', i, " eme athlete") ;
              Lire(Num, Nom, H, M, S) ;
         Fait
     Fait

     // Calcul de la moyenne de tous les athletes
     moy <-- 0;
     Pour i <-- 1 a N  Faire
         moy <-- moy + Transform(T[i].Res);
     Fait
     moy <-- moy Div N;

     // affichage de la liste des athletes dont le temps est <= moyenne
     Ecrire("Liste des athletes dont le temps est <= Moyenne des athletes");
     Pour i <-- 1 a N  Faire
         Avec T[i] Faire
            Si (Transform(Res) <= moy) alors
                Ecrire(Num, Nom, " Temps(", Res.H, Res.M, Res.S, ")");
            Fsi
         Fait
     Fait
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 petit 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 le nombre Zs-Zp et son module.

1-

```
Type
      Complexe=Enregistrement
          a, b : Reel ;
       Fin;
```

2-

```
Fonction      ReelZ(Complexe z) : Reel
Debut
    Retourner z.a;
Fin;

Fonction      ImagZ(Complexe z) : Reel
Debut
    Retourner z.b;
Fin;

Fonction      MODule(Complexe z) : Reel
Debut
    Retourner (sqrt(z.a * z.a + z.b * z.b));
Fin;
```

3-

```
Fonction      SommeZ(Complexe z1, z2) : Complexe
Var
   z : Complexe;
Debut
   z.a <--- z1.a + z2.a;
   z.b <--- z1.b + z2.b;

   Retourner  z;
Fin;

Fonction      DiffZ(Complexe z1, z2) : Complexe
Var
   z : Complexe;
Debut
   z.a <--- z1.a - z2.a;
   z.b <--- z1.b - z2.b;

   Retourner  z;
Fin;

Fonction      ProdZ(Complexe z1, z2) : Complexe
Var
   z : Complexe;
Debut
```

```
    z.a <--- z1.a * z2.a - z1.b * z2.b;
    z.b <--- z1.a * z2.b + z1.b * z2.a;

    Retourner  z;
Fin;
```

4-

```
Fonction       ConjZ(Complexe z) : Complexe
Var
    zc : Complexe;
Debut
    zc.a <--- z.a;
    zc.b <--- -z.b;

    Retourner  zc;
Fin;
```

5-

```
Fonction       EgaleZ(Complexe z1, z2) :  Booleen
Debut
    Si (( z1.a = z2.a) ET (z1.b = z2.b)) alors
            Retourner  Vrai;
    Sinon
            Retourner  Faux;
    Fsi
Fin;
```

6-

```
Procedure     EcrireZ(E/  Complexe z)
Debut
     Si (z.b = 0)  alors
           Ecrire(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;

Algorithme     NombresComplexes;
Type
   Complexe=Enregistrement
           a, b : Reel ;
   Fin;

Var
  TC: Tableau[1..100] de Complexe;
  Z0, Zs, Zp, Zmin, Zd, Zconj : Complexe;
  i, N: Entier;
  Min : Reel;
  Trouve : Booleen;

/* Declaration des A.P. */
```

```
 ...

Debut
  Repeter
    Ecrire("Donner N Compris entre 1 et 100");
    Lire(N);
  Jusqu'a ((N>=1) ET (N<=100));

  Pour i <--- 1 a N
  Faire
      Lire(TC[i].a, TC[i].b);
  Fait

  /* Recherche de l'element ayant le plus petit module */
  Zmin <--- TC[1];
  Min <--- MODule(Zmin);
  Pour i <--- 2 a N
  Faire
     Si (MODule(TC[i]) < Min) alors
          Zmin <--- TC[i];
          Min   <---  MODule(TC[i]);
     Fsi
  Fait

  Ecrire("Element ayant le plus petit module:");
  EcrireZ(Zmin);

  /* Verifier l'existence du conjugue de zMin */
   Trouve <--- Faux;
   Zconj <--- ConjZ(Zmin);
   i <--- 1;
   Tant Que ( (i<=N) ET (Trouve = Faux))
   Faire
       Si (EgaleZ(TC[i], Zconj) = Vrai)  alors
             Trouve <--- Vrai;
       Sinon
            i  <--- i + 1;
       Fsi
   Fait

   Si (Trouve) alors
        Ecrire("Le conjugue de Zmin Existe dans TC");
   Sinon
        Ecrire("Le conjugue de Zmin N'Existe PAS dans TC");
   Fsi

   /* Calcul de la somme et du produit des elements non nuls du tableau TC */
   Z0.a <--- 0;
   Z0.b <--- 0;
   Zs <--- Z0;
   Zp.a <--- 1;
   Zp.b <--- 0;

   Pour i <--- 1 a N
   Faire
       Si (EgaleZ(TC[i], Z0) = Faux)   alors
            Zs <--- SommeZ(Zs, TC[i]);
            Zp <--- ProdZ(Zp, TC[i]);
       Fsi
   Fait
```

```
  Ecrire("Somme des elements non nuls du tableau TC:");
  EcrireZ(Zs);
  Ecrire("Produit des elements non nuls du tableau TC:");
  EcrireZ(Zp);

  /* Calcul et affichage de la différence entre Zs et Zp et de son module */
  Zd <--- DiffZ(Zs, Zp);
  Ecrire("Zd = ");
  EcrireZ(Zd);
  Ecrire("Module de Zd = ", MODule(Zd));

Fin.
```

Exercice 3 :

Soit TDate un type date composé des champs entiers JJ, MM, AA. Ecrire les APs suivantes :
- CompareD : compare deux dates D1 et D2 (-1 , 0 , 1 pour inférieur, égale, et supérieur).
- BIS : vérifie si une année est bissextile.
- Valide : vérifie la validité d'une date.
- Diffjour : calcule la différence en jours entre deux dates.

Soit TD un tableau de N dates (N≤100). Ecrire un algorithme permettant de :
- Trier ce tableau dans l'ordre croissant des dates.
- Calculer la différence minimale en nombre de jour entre ces dates.

```
TYPE

 TDate = Enregistrement
   JJ, MM, AA : Entier;
 Fin;

/* CompareD: Cette fonction compare 2 date D1 et D2 et retourne:
  1 si D1 > D2
  -1 si D1 < D2
  0 si D1 = D2
*/
Fonction    compareD(D1, D2: TDate): Entier
Var
  D1CompareD2 : Entier;

Debut
   Si (D1.AA > D2.AA)  alors
      D1CompareD2 <--- 1;
   Sinon
     Si (D1.AA < D2.AA)  alors
        D1CompareD2 <--- -1;
     Sinon
       Si (D1.MM > D2.MM)  alors
          D1CompareD2 <--- 1;
       Sinon
         Si (D1.MM < D2.MM)  alors
            D1CompareD2 <--- -1;
         Sinon
           Si (D1.JJ > D2.JJ)  alors
              D1CompareD2 <--- 1;
           Sinon
             Si (D1.JJ < D2.JJ)  alors
                D1CompareD2 <--- -1;
             Sinon
                D1CompareD2 <--- 0;
             Fsi
           Fsi
```

```
        Fsi
      Fsi
    Fsi
  Fsi

  Retourner D1CompareD2;
Fin;

/*
BIS: Fonction qui retourne Vrai si une annee est bissextile; sinon retourne Faux
Une annee est bissextile si annee est divisible par 4 et n'est pas divisible par 100;
ou bien est divisible par 400
*/
Fonction    BIS(aa: Entier): Booleen
Debut
 Si ( ((aa Mod 4 = 0) ET (aa Mod 100 <> 0)) OU (aa Mod 400 = 0) )
     Retourner Vrai;
 Sinon
     Retourner Faux;
 Fsi
Fin;

/* Valide: Fonction qui retourne Vrai si une date est valide et retourne Faux dans le cas
contraire */
Fonction     Valide(D: TDate) : Booleen
Var
  DValide : Booleen;
Debut

  /* on suppose au debut que la date n'est pas valide et on va la mettre a Vrai uniquement
dans le
     cas ou la date est valide.
   */
  DValide <--- Faux;

  Si (D.AA >= 0)  alors
     Cas (D.MM)  Vaut
       /* les mois qui ont 31 jours janv, mars, mai, juill, aout, oct et dec */
       1, 3, 5, 7, 8, 10, 12:
          Si ( (D.JJ >= 1) ET (D.JJ <= 31) )  alors
             DValide <--- Vrai;
          Fsi

       /* les mois qui ont 30 jours avr, juin, sept, nov */
       4, 6, 9, 11:
          Si ( (D.JJ >= 1) ET (D.JJ <= 30) )  alors
             DValide <--- Vrai;
          Fsi

       /* mois de fevrier: si annee bissextile alors mois de 29 jours sinon mois de 28 jours */
       2:
          Si (BIS(D.AA))  alors
             Si ( (D.JJ >= 1) ET (D.JJ <= 29) )  alors
                DValide <--- Vrai
             Fsi
          Sinon
             Si ( (D.JJ >= 1) ET (D.JJ <= 28) )  alors
               DValide <--- Vrai;
             Fsi
           Fsi
```

```
            Fsi
        FCas
    Fsi

    retourner DValide;
Fin;

/*
NbJoursAnnee: Fonction qui retourne le nombre de jours du 01-01-d.aa au d.jj-d.mm-d.aa
On a besoin de cette fonction dans la fonction Diffjour.
Exemple: Nbj du 01-01-2019 au 15-07-2019
--------------|----------|---------|--------|-------|--------|----------|------------------------------
          01-01-2019      02          03          05        06         07     15-07-2019
Nbj du 01-01-2019 au 15-07-2019 = 15j (mois 07) + Nbj mois(01 + 02 + 03 + 04 + 05 + 06)
*/
Fonction    NbJoursAnnee(D: TDate):Entier
Var
    i, nbj : Entier;

Debut
    /* initialiser nbj au nombre de jours du dernier mois */
    nbj <--- D.JJ;

    Pour i <--- 1 a (D.MM - 1)
    Faire
        cas i Vaut
          /* les mois qui ont 31 jours janv, mars, mai, juill, aout, oct et dec  ajouter 31 jours*/
          1, 3, 5, 7, 8, 10, 12:
              nbj <--- nbj + 31;

          /* les mois qui ont 30 jours avr, sept, nov ajouter 30 jours*/
          4, 6, 9, 11:
              nbj <--- nbj + 30;

          /* mois de fevrier: si annee bissextile alors mois de 29 jours sinon mois de 28 jours */
          2:
              Si (BIS(D.AA))  alors
                   nbj <--- nbj + 29;
              Sinon
                   nbj <--- nbj + 28;
              Fsi
        FCas
    Fait

    retourner nbj;
Fin;

/* Diffjour: Fonction qui calcule la difference en nombre de jours entre 2 dates D1 et D2
Exemple: Nbj entre D1 = 14-05-2002 et D2 = 17-08-2006
--------------|----------|------------------|-------|--------|----------|------------|-------------------
          01-01-2002  14-05-2002           2003   2004   2005       2006       17-08-2006
                              ------------------------------------------------------------14-05-2002 au 17-08-
2006
                                                                                  ------------01-01-2006
au 17-08-2006
                    ------------------------------------------------------------01-01-2002 au 31-12-2005
                    ------------01-01-2002 au 14-05-2002
Nbj du 14-05-2002 au 17-08-2006 =
Nbj 01-01-2006 au 17-08-2006 + Nbj 01-01-2002 au 31-12-2005 - Nbj du 01-01-2002 au 14-
05-2002
```

```
*/
Fonction    Diffjour(D1, D2: TDate) : Entier
Var
   i, nbj : Entier;
   d : TDate;

Debut

   /* mettre dans D1 la plus petite des 2 dates D1 et D2  */
   Si (compareD(D1, D2) = 1) alors
      /* permuter d1 et d2 si D1 > D2 */
      D <--- D1;
      D1 <--- D2;
      D2 <--- D;
   Fsi

   nbj <---  NbJoursAnnee(D2);

   /* ajouter les nombres de jours a partir de la 1ere annee (D1.AA) jusqu'a l'avant derniere
annee (D2.AA - 1) */

   Pour i <--- D1.AA a  (D2.AA - 1)
   Faire
      Si (BIS(i))  alors
         /* Si annee est bissextile ajouter 366 jours */
         nbj <--- nbj + 366;
      Sinon
         /* annee non bissextile ajouter 365 jours */
         nbj <--- nbj + 365;
      Fsi
   Fait

   /* retrancher le nombre de jours qu'il y'a dans la 1ere annee (d1.aa)  */
   nbj <--- nbj - NbJoursAnnee(D1);

   retourner nbj;
Fin;


Algorithme     TableauDates;
TYPE
  TDate = Enregistrement
     JJ, MM, AA : Entier;
  Fin;

Var
  TD: Tableau[1..100] de TDate;
  i, J, nb, minNbJr, iMin: Entier;
  D : TDate;

  /* Declaration des A.P. */

  ...

Debut
  Repeter
     Ecrire("Donner N Compris entre 1 et 100");
     Lire(N);
  Jusqu'a ((N>=1) ET (N<=100));
```

```
Pour i <--- 1 a N
Faire
    Repeter
      Ecrire("Donner Valide Date(JJ-MM-AA) de la ", i, " eme Date");
      Avec TD[i] Faire
        Lire(JJ, MM, AA);
      Fait
    Jusqu'a (Valide(TD[i]));
Fait

/* Tri par Selection du tableau TD */
Pour i <--- 1 a (N - 1)
Faire
    /* Chercher la  date la plus petite du tableau de i a N */
    iMin <--- i;
    Pour J <--- (i + 1) a N
    Faire
      Si (CompareD(TD[J], TD[iMin])  = - 1) alors
        iMin <--- J;
      Fsi
    Fait

    Si (i <> iMin) alors
      /* Permuter TD[i] et TD[iMin] */
      D <--- TD[i];
      TD[i] <--- TD[iMin];
      TD[iMin] <--- D;
     Fsi
  Fait

  /* Calculer la différence minimale en nombre de jour entre ces dates
     Comparer la difference entre toutes les 2 dates qui se suivent et chercher le min de ces
differences
   */
  Si (N = 1) alors
    Ecrire("Il y'a une seule Date") ;
  Sinon
    minNbJr <--- Diffjour(TD[1], TD[2]);
    Pour i <--- 2 a (N - 1)
    Faire
      nb <--- Diffjour(TD[i], TD[i + 1]);
      Si (nb < minNbJ) alors
        minNbJr <--- nb;
      Fsi
    Fait

    Ecrire("Différence minimale en nombre de jour = ", minNbJr);
  Fsi
Fin.
```

Exercice 4 :

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"
Resultat: 3 - 6- 8-12 4

```
TYPE
  E = Enregistrement
    T : Tableau[1..100] de Entier;
    N : Entier;
  Fin;

// T a 100 elements on vas declarer M comme une chaine de 200 caracteres pour avoir au
// maximum 100 occurrences de "ab" dans M ("ababab....ab")
Fonction   Rechercher(M: Chaine[200]) : E
Var
  res : E;
  i, t : Entier;

Debut
    res.N <--- 0;
    t <--- Taille(M);
    i <--- 1;
    Tant Que (i < t)
    Faire
        Si ((M[i] = 'a') ET (M[i+1] = 'b'))  alors
            res.N <--- res.N + 1;
            res.T[res.N] <--- i;
            i <--- i + 2;
        Sinon
            i <--- i + 1;
        Fsi
    Fait

    Retourner res;
Fin;
```

Exercice 5 :

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 : TDate ;
      Residence : TAdresse ;
   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 AN donnée.

3- Afficher les noms et les dates de naissance des habitants de la ville V d'une wilaya W.

4- Afficher le nombre d'habitants par wilaya.

```
Algorithme       TabHabitant ;
Type
    Date = Enregistrement
         Jour, mois, année : entier ;
    Fin;
    Adresse = Enregistrement
         Numero : entier ;
         Rue : chaine [50] ;
         Ville : chaine [20] ;
         Wilaya : chaine [20] ;
         Cw : entier ;  /* Code Wilaya */
     Fin;
     Habitant = Enregistrement
         Nom, prenom : chaine [20] ;
         Date_naiss : Date ;
         Residence : Adresse ;
    Fin;

Var
   T :Tableau[1..100] de Habitant ;
   CptW :Tableau[1..58] de entier ;
   V, W : Chaine[20];
   i,N,An :entier ;

Debut
   Repeter
       Ecrire("Donner N Compris entre 1 et 100");
       Lire(N);
  Jusqu'a ((N>=1) ET (N<=100));

  /* 1- Remplir tableau T de N habitants (N≤100). */
  Pour i <--- 1  a  N
  Faire
        Avec T[i]  Faire
              Lire(Nom, Prenom);
              Lire(Date_naiss.Jour, Date_naiss.mois, Date_naiss.annee);
              Lire(Residence.Numero,  Residence.Rue, Residence.Ville, Residence.Wilaya,
Residence.Cw);
        Fait
  Fait

  /* 2- Afficher à partir de T les adresses des habitants nes avant une année de naissance
donnee. */
   Ecrire("Donner une Annee") ;
   Lire(An) ;
   Pour i <--- 1 a N
   Faire
        Avec T[i].Date_naiss, T[i].Residence   Faire
            Si (Annee<An) Alors
                  Ecrire(Numero,Rue,Ville,Wilaya)
            Fsi;
        Fait ;
    Fait;

  /* 3- Afficher les noms et les dates de naissance des habitants de la ville de V de la wilaya
W. */
    Ecrire("Donner Ville V et Wilaya W");
    Lire(V, W);

    Ecrire("Debut de Liste des Habitants de la ville ", V, " et de la Wilaya ", W);
```

```
   Pour i <--- 1  a  N
   Faire
       Avec T[i],T[i].Date_naiss, T[i].Residence   Faire
             Si ((Ville = V) et (Wilaya = W))  Alors
                   Ecrire(Nom,Prenom,Jour,'/',Mois,'/',Annee)
             Fsi
       Fait
   Fait
   Ecrire("Fin de Liste");

  /* 4- Editer le nombre d'habitants par wilaya.
        On utilise pour cela un tableau de 58 elements CptW ou chaque element CptW[i] est
le compteur
        de la wilaya dont le code de la wilaya CW est egale a I. Par exemple le
        compteur de la wilaya d'alger est CptW[16] car le CW de la wilaya d'alger est 16.
Donc on utilise
        le champ CW comme indice dans CptW pour acceder au compteur correspondant de
la wilaya.
        On suppose bien sur ici que les codes de wilaya tels que donnes par l'utilisateur sont
corrects (c.a.d compris entre 1 et 58).
     */

   /* Initialiser tous les compteurs a 0 */.
   Pour i <--- 1  a  58
   Faire
       CptW[i] <--- 0 ;
   Fait

   /* Parcourir tous le tableau T et incrementer le compteur de la wilaya de residence de
l'habitant T[I].
      L'indice du compteur de la wilaya est le champ CW de la residence de l'habitant.
     */
   Pour i <--- 1  a  N
   Faire
        CptW[T[i].ReSidence.CW]<--- CptW[T[i].Residence.CW] + 1 ;
   Fait

   /*affichage */
   Pour i <--- 1 a 58
   Faire
        Ecrire("Wilaya Code: ", i ,  " Nombre  d'Habitants: ",CptW[i]) ;
   Fait
Fin
```

Exercice 6 :

Considérons les types enregistrement suivant :

```
Type
  TModule = Enregitrement
    Nom : chaine[10] ;
    Note :reel ;
    Coef : entier ;
  Fin ;
  TEtudiant = Enregistrement
    Matricule : entier ;
    Nom, Prenom : chaine [20] ;
    M1, M2, M3 : TModule ;
    Moyenne : réel ;
  Fin;
```

Soit T un tableau d'au plus 100 étudiants. Ecrire un algorithme permettant de remplir le tableau T,
calculer la moyenne de chaque étudiant puis recopier tous les étudiants admis dans un tableau ADMIS de type TEtudiant. Un étudiant est admis si sa moyenne est supérieure ou égale 10.

```
Algorithme   TabEtudiantsAdmis;
Type
  TModule = Enregitrement
     Nom : chaine[10] ;
     Note :reel ;
     Coef : entier ;
  Fin ;

  TEtudiant = Enregistrement
     Matricule : entier ;
     Nom, Prenom : chaine [20] ;
     M1, M2, M3 : TModule ;
     Moyenne : réel ;
  Fin;

Var
  T, ADMIS: Tableau[1..100] de TEtudiant;
  i, J, N: Entier;

Debut
  Repeter
       Ecrire("Donner N Compris entre 1 et 100");
       Lire(N);
  Jusqu'a ((N>=1) ET (N<=100));

  /* Remplir tableau T de N etudiants (N<=100). */
  Pour i <--- 1  a  N
  Faire
       Avec T[i]  Faire
            Ecrire("Donner Matricule, Nom et Prenom de Etudiant ...");
            Lire(Matricule, Nom, Prenom);
            Donner("Nom, Note et Coefficient de Module 1");
            Lire(M1.Nom, M1.Note, M1.Coef);
            Donner("Nom, Note et Coefficient de Module 2");
            Lire(M2.Nom, M2.Note, M2.Coef);
            Donner("Nom, Note et Coefficient de Module 3");
            Lire(M3.Nom, M3.Note, M3.Coef);
       Fait
  Fait

  /* Calcul de la moyenne de chaque etudiant */
  Pour i <--- 1  a  N
  Faire
     T[i].Moyenne <--- ((T[i].M1.Note * T[i].M1.Coef) + (T[i].M2.Note * T[i].M2.Coef) +
(T[i].M3.Note T[i].M3.Coef) ) / (T[i].M1.Coef + T[i].M2.Coef + T[i].M3.Coef);
  Fait

  /* Mettre dans ADMIS les etudiants admis */
  J <--- 0;
  Pour i <--- 1  a  N
  Faire
     Si(T[i].Moyenne >= 10) alors
        J <--- J + 1;
```

```
        ADMIS[J] <--- T[i];
      Fsi
    Fait

    Si (J = 0) alors
      Ecrire("Tableau ADMIS est vide");
    Sinon
      Ecrire("Liste des etudiants admis:");
      Pour i <--- 1 a J
      Faire
        Avec T[i] Faire
          Ecrire(Matricule, Nom, Prenom);
        Fait
      Fait
    Fsi
  Fin.
```