**Université de Bejaia**
**Faculté des sciences exactes**
**Département M/MI**
**1ère année MI**
**Module PSSD**

**Nom :**
**Prénom :**
**Groupe :**
**Emargement:**
**Lundi 21/12/2015, Durée :1H30MN**

**Exercice 01 :**

```
algorithme exo_01 ;
    Type
        contact = enregistrement
                    nom, pre: chaine[30] ;
                    numt:entier
                    adr: chaine[30]  ;
                    Fin ;
        tabc=tableau[1..200] de contact ;

   Var     tac :tabc ;nb :entier ; adrr: chaine[30] ;numtt :entier ;
    Procedure      agendat_saisir (var TC: tabc) ;
        Var              i: entier ;   rep :caractere ;
          Debut
          |      nb ←0;// une variable globale
          |     repeter
          |     |          avec TC[i] faire
          |     |          |       lire ( nom, pre, numt, adr);
          |     |          finavec ;
          |     |        ecrire('voulez vous ajouter un contact ?O/N') ;
          |     |          lire(rep) ;
          |     |          nb ←nb+1 ; //nombre des enregistrements saisis, vous pouvez le calculer ailleurs.
          |     |jusqu'à (rep='N');
          |fin ;

    Procedure      cherche_agendat (var TC: tabc ; nc: entier ; adrr1: chaine[30]) ;
        Var              i: entier ;     trouve:booleen;
          Debut
          |     i← 1;
          |     trouv ← faux  ;
          |     tantque    ((i≤nc) et non trouv )  faire
          |     |          avec TC[i] faire
          |     |          |  Si    (adr= adrr1)  alors
          |     |          | trouv← vrai ;
          |     |          |sinon
          |     |          |       i ← i+1 ;
          |     |          |finsi ;
          |     |fintantque ;
          |     si (trouve=vrai) alors ecrire (nom, pre, numt) ;
          |     |sinon      ecrire('l''adresse électronique n''existe pas') ;
          |     |finsi;
          |fin ;
```

```
Procedure     modif_agendat (var TC: tabc ; nc: entier ; numtt1:entier) ;
      Var             i: entier ;     trouve:booleen;
        Debut
        |     i← 1;
        |     trouv ← faux  ;
        |     tantque    ((i≤nc) et non trouv )  faire
        |     |          avec TC[i] faire
        |     |          |  Si    (numt= numtt1)  alors
        |     |          | trouv← vrai ;
        |     |          |sinon
        |     |          |       i ← i+1 ;
        |     |          |finsi ;
        |     |fintantque ;
        |     si (trouve=vrai) alors lire (nom, pre, numt,adr) ;
        |     |sinon      ecrire('Ce numéro de téléphone n''existe pas') ;
        |     |finsi;
        |fin ;

        Debut
        |     agendat_saisir (tac) ;
        |     lire (adrr) ;
        |     cherche_agendat (tac, nb,adrr) ;
        |     lire (numtt) ;
        |     modif_agendat (tac,nb, numtt) ;
        | fin .
```

**Exercice 2:**

```
algorithme exo_02;
        Type    pointeur = ↑contact ;
             contact = enregistrement
                              nom, pre: chaine[30] ;
                              numt:entier
                              adr: chaine[30]  ;
                              Suivant : pointeur ;
                               Fin ;
       Var      L :pointeur ; numtt,N :entier ; S :chaine[3] ;

   Procedure       creation (var liste : pointeur) ;
              Var                P   : pointeur ; rep :chaine[3] ;

                 Debut
                 |     liste← Nil ;
                 |     repeter
                 |     |          allouer (P)   ;
                 |     |   avec P↑ faire
                 |     |  |        lire (nom, pre, numt,adr)   ;
                 |     |  |        suivant ←liste ;
                 |     |   finavec ;
                 |     |          liste←  P ;
                 |     |    ecrire('voulez vous rajouter un médicament ?oui/non') ;
                 |     |    lire (rep) ;
                 |     |jusqu'à (rep='non');
                 |fin ;

    procedure    ajouter(var liste : pointeur) ;
              Var                P   : pointeur ;
                 Debut
                 |     P← liste;
                 |     tantque    (P↑.suivant <> Nil)    faire
                 |     |          P ← P↑.suivant ;
                 |     |fintantque ;
                 |
                 |     allouer (Q)   ;
                 |     avec Q↑ faire
                 |     |        lire (nom, pre, numt,adr)   ;
                 |     |        suivant ←Nil ;
                 |     |finavec ;
                 |     P↑.suivant ←Q ;
                 |fin ;
```

```
procedure     supprimer (var liste : pointeur ; numtt1 :entier) ;
      Var             P, R   : pointeur ;
        Debut
        |     P ← liste;
        |   si (P↑.numt =numtt1)  alors
        |   |                              liste ← liste↑.suivant ;
        |   |                              liberer (P) ;
        |   |sinon
        |   |  R← P↑.suivant ;
        |   | Tantque   (R <> NiL)    faire
        |   |   |    si (R↑.numt =numtt1)  alors
        |   |   |    |                       si (R↑.suivant<>Nil) alors
        |   |   |    |                       |                  P↑.suivant ← R↑.suivant;
        |   |   |    |                       |                  liberer (R ) ;
        |   |   |    |                       |sinon
        |   |   |    |                       |       P↑.suivant ← Nil;
        |   |   |    |                       |       liberer (R ) ;
        |   |   |    |                       finsi ;
        |   |   |    |finsi ;
        |   |   |         P ← R;
        |   |   |         R ← R↑.suivant;
        |   |  |fintantque ;
        |  |finsi ;
        |fin ;

  procedure     afficher (liste : pointeur ; S1 :chaine[3]) ;
      Var             P   : pointeur ;
        Debut
        |     P← liste;
        |     tantque   (P<> Nil)   faire
        |     |    avec P↑ faire
        |     |    |  Convch( numt,ch)   ; //convertir le numéro de téléphone (entier) en une chaine ch
        |     |    |  res ← position(S1,ch);// chercher la position de S1 dans ch
        |     |    |    Si   (res<>0)  alors
        |     |    |    | ecrire (nom,pre,numt,adr)   ;
        |     |    |    |finsi ;
        |     |    finavec ;
        |     |     P ← P↑.suivant ;
        |     |fintantque ;
        |fin ;

Debut
|     creation (L) ;
|     ajouter (L) ;
|     lire (numtt) ; supprimer (L, numtt) ;
|     repeater  lire(N) jusqu'a ((N≥100) et (N≤999));
|     Convch(N,S) ; afficher (L,S) ;
|fin.
```

**Exercice 3:**

```
algorithme     chercher_min ;
type     pile= ↑ nœud   ;
type     nœud  = enregistrement
           | info :entier ;
           | suivant : pile ;
           |fin ;
var   p :pile ; E, Min, i : entier ;
debut
|        vider_pile (p) ;
|        pour   i     allantde     1     jusqu'à   10   faire
|        |    lire (E) ;
|        |    empiler (p,E) ;
|        | finpour;
|        Min ← sommet(p) ;
|        tantque    (pile_vide(p)=faux)   faire
|        |   depiler(p) ;
|        |    Si (Min> sommet(p)) alors
|        |    |  Min← sommet(p) ;
|        |    |finsi ;
|        | fintantque;
|         ecrire (Min) ;
|fin.
```