CONTROLE D'ALGORITHMIQUE ET PROGRAMMATION

Durée : 1h30 Seule la feuille de syntaxe ADA est autorisée

Soignez la présentation ; Ecrivez lisiblement ; Commentez.

Barème indicatif : Exercice 1 : 5 points ; Exercice 2 : 5 points ; Exercice 3 : 10 points

Toutes les réponses aux exercices doivent être données dans les fichiers indiqués dans les énoncés. Vous perdrez du temps à essayer de réécrire les questions de l'exercice 1 pour les tester… Le sous programme de l'exercice 2 ne doit être ni compilé ni exécuté. Les programmes de l'exercice 3 doivent être compilés et exécutés. Lire attentivement les consignes de compilation et d'exécution données en début d'exercice 3. La question 2 de l'exercice 3 peut être traitée sans avoir traitée la question 1.
Pensez également à mettre votre nom, votre prénom et votre groupe systématiquement dans tous les fichiers que vous complétez. Le répertoire de travail est "~/interroADA".

EXERCICE 1 : Toutes notions 15'
Quatre programmes vous sont proposés. Pour les quatre, les acteurs suivants sont déclarés :

```
with Ada.Text_Io, Ada.Integer_Text_Io ;
use Ada.Text_Io, Ada.Integer_Text_Io ;
```

Préciser le résultat obtenu à l'exécution dans le fichier exercice1.adb.

```
procedure UN is

  type Element ;
  type P_Element is access Element ;
  type Element is record
    Info : Integer ;
    Suiv : P_Element ;
  end record ;

  procedure Afficher_Liste(L :in out  P_Element) is
  begin
    if L /= null then
      put(L.all.Info);
      Afficher_Liste(L.all.Suiv);
    end if;
  end Afficher_Liste ;


  procedure Creer1_Liste(L :in out  P_Element ; I : in Integer) is
     aux : P_Element ;
  begin
    if I /= 0 then
       aux := new Element'(I,null);
       L := Aux ;
       Creer1_Liste(L,I-1);
    end if;
  end Creer1_Liste ;

  procedure Creer2_Liste(L :in out  P_Element ; I : in Integer) is
     aux : P_Element ;
  begin
    if I /= 0 then
       Creer2_Liste(L,I-1);
       aux := new Element'(I,L);
       L := Aux ;
    end if;
  end Creer2_Liste ;

  Ma_Liste1,Ma_Liste2 : P_Element := null ;

begin
  Creer1_liste(Ma_liste1,4);
  Creer2_liste(Ma_liste2,4);
  put_line("Ma liste 1");
   Afficher_Liste(Ma_Liste1) ;
   new_line;
  put_line("Ma liste 2");
   Afficher_Liste(Ma_Liste2) ;
end UN ;
```

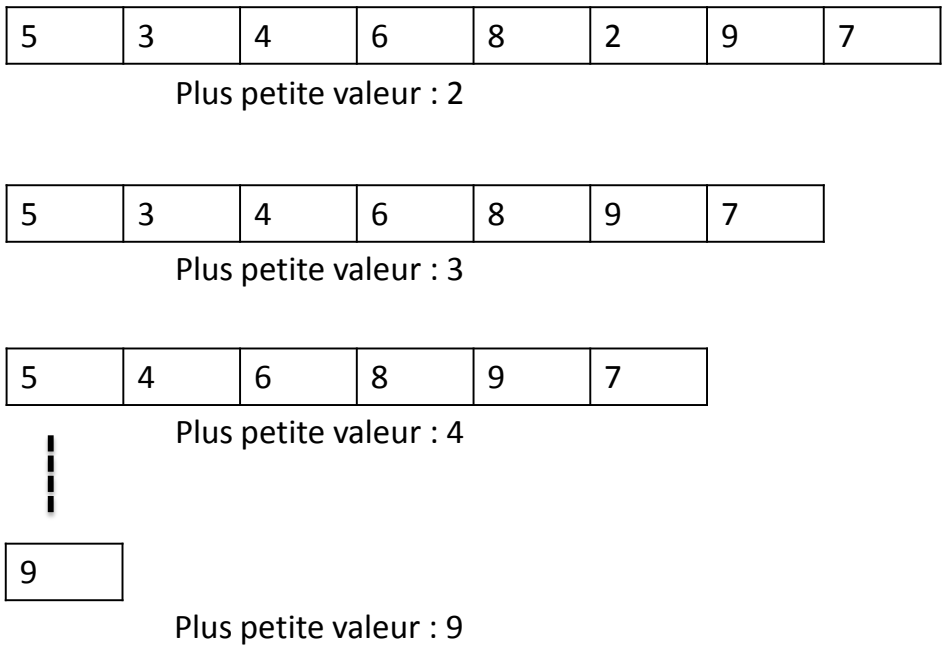
```
procedure DEUX  is

  type Element ;
  type P_Element is access Element ;
  type Element is record
    Info : Integer ;
    Suiv : P_Element ;
  end record ;

  procedure Creer (L : out P_Element) is
    Aux:P_Element ;
  Begin
    L := new Element'(1,null);
    Aux := L ;
    for I in 2..4 loop
      Aux.all.suiv := new Element'(I,null);
      Aux:=Aux.all.suiv;
    end loop;
  end Creer;

  procedure Proc1 (L : in P_Element) is
     Aux:P_Element := L ;
  begin
    Aux.all.suiv := null ;
  End Proc1 ;

procedure Afficher_Liste(L :in P_Element) is
  begin
    if L /= null then
      put(L.all.Info);
      Afficher_Liste(L.all.Suiv);
    end if;
  end Afficher_Liste ;

  Ma_Liste1,Ma_Liste2 : P_Element := null ;

begin
  Creer(Ma_Liste1) ;
  Ma_liste2 := Ma_liste1 ;
  Proc1(Ma_liste2) ;
  put_line("Ma liste 1");
  Afficher_Liste (Ma_Liste1) ;
  New_Line;
  put_line("Ma liste 2");
  Afficher_Liste (Ma_Liste2) ;


end DEUX ;
```

```
procedure TROIS is

  type T_Tab is array (Integer range <>) of Integer;


  procedure Proc1 (T : in T_Tab) is
  begin
    for i  in 1..2 loop
      put(T(I));
    end loop;
  end Proc1 ;

  Mon_Tab : T_Tab(1..4) := (40,30,20,10) ;

begin
  put("Afficher T") ;
  proc1(Mon_Tab) ;
  new_line;
  put("Afficher T(3..4)") ;
  proc1(Mon_Tab(3..4));
  new_line;
  put("Afficher T(1..2)") ;
  proc1(Mon_Tab(1..2));
  new_line;
end TROIS ;
```

```
procedure QUATRE is

  type T_Tab_NC is array (Integer range <>) of Integer;

  procedure Afficher_1 (V : in T_Tab_NC) is
  begin
    if  V'Length >= 1  then
      Put(V(V'Last));
      Afficher_1(V(V'First..V'Last-1)) ;
    end if;
  end Afficher_1;

  procedure Afficher_2 (V : in T_Tab_NC) is
  begin
    if V'Length >= 1  then
       Put (V(V'Last));
      Afficher_2(V(V'First..V'Last-1)) ;
      Put (V(V'Last));
    end if;
  end Afficher_2;

  procedure Afficher_3 (V : in out T_Tab_NC) is
  begin
    if V'Length >= 1  then
      Put (V(V'Last));
      Afficher_3 (V(V'First+1..V'Last)) ;
      Put (V(V'Last));
    end if;
  end Afficher_3;

  Mon_Tab : T_Tab_NC(1..4) := (10, 20, 30, 40) ;

begin
  put ("Affichage 1 :");
  Afficher_1 (Mon_Tab);new_line;
  put ("Affichage 2 :");
  Afficher_2 (Mon_Tab);new_line;
  put ("Affichage 3 :");
  Afficher_3 (Mon_Tab);New_Line;
end QUATRE ;
```

EXERCICE 2 : Tableau et Récursivité. 15'

On souhaite faire un programme qui affiche la plus petite valeur *v1* d'un tableau puis la plus petite valeur *v2* d'un tableau privé de la valeur *v1* et ainsi de suite jusqu'à avoir affiché toutes les valeurs du tableau par ordre croissant.



Proposez **<u>un algorithme récursif</u>** *AfficherOrdreCroissant* permettant d'afficher les plus petites valeurs successives d'un tableau initial.

<u>Compléter l'entête et le corps du sous-programme dans le fichier exercice2.adb.</u>

EXERCICE 3 : Mise au point d'un programme. 60'

- Vous êtes connecté sur un compte spécifique à l'examen, vous devez utiliser la console pour :
  - lancer l'éditeur *emacs*, par ex. *emacs toto.adb&*
  - compiler vos programmes à l'aide de la commande *gnatmake*, par ex. *gnatmake toto.adb*
  - exécuter vos programmes à l'aide de la commande *run*, par ex. *run toto*
- Respectez les noms de fichier imposés dans le sujet. Tout autre fichier ne sera pas corrigé par le correcteur… Votre répertoire de travail est votre *home directory*.
- Avant de partir :
  - sauvegardez votre programme
  - déconnectez-vous.

Votre programme sera directement envoyé par *email* aux enseignants à l'heure de clôture des comptes. Il est donc impératif de quitter votre session avant la sauvegarde, les fichiers encore ouverts au moment de la sauvegarde peuvent être mal enregistrés.

Question 1 – Compléter le fichier *exercice3q1.adb*

Lire attentivement le contenu de ce fichier, notamment, vous retrouverez la déclaration du type Liste suivante :

```
Type Element;

Type P_Element is access Element;

Type Element is record
   Info : Integer ;
   Suiv : P_Element ;
end record ;

Type Liste is record
    Debut : P_Element ;
    Fin : P_Element ;
end record;
```

Ainsi que la déclaration et le corps d'un sous-programme d'affichage d'éléments chaînés entre eux que vous ne devez pas modifier.

1. Ecrire un sous-programme *GenererListe* qui saisit une suite de valeurs entières fournies par l'utilisateur du programme et les range au fur et à mesure de la saisie dans une liste chaînée. Les valeurs inférieures à un seuil donné sont rangées en début de liste ; les valeurs supérieures ou égales sont rangées en fin de liste. La saisie s'arrête lorsque l'utilisateur entre la valeur 0 qui n'est pas rangée dans la liste et ce sous-programme retourne en plus de la liste le nombre d'éléments saisis.

Remarque : l'ordre de rangement des valeurs n'a pas d'importance dès lors que la consigne sur le seuil est respectée.

2. Ecrire le programme principal qui teste votre sous-programme en affichant la liste et le nombre d'éléments de la liste.

Exemples d'exécution (les caractères gras sont affichés par le programme)

Si l'utilisateur fournit :

- 2 comme valeur du seuil ;
- successivement les valeurs : 4 -5 8 2 6 -3 -9 1 0

On obtiendra lors de l'affichage final :

```
Valeurs contenues dans la liste : 1 -9 -3 -5  4  8  2  6
```

Question 2 – Compléter le fichier *exercice3q2.adb*

Lire attentivement le contenu de ce fichier, notamment, vous retrouverez la déclaration du type T_Tab suivante :

```
type T_Tab_NC is array(Integer  range <>) of Integer;
```

Ainsi que la déclaration et le corps d'un sous-programme d'affichage d'éléments d'un tableau que vous ne devez pas modifier.

> Si votre sous programme *GenererListe* de la question 1 fonctionne correctement copiez le et remplacez celui fournit !!! Sinon vous pouvez utiliser celui fournit…

1. Ecrire un sous-programme qui génère un tableau contenant les valeurs strictement inférieures à un seuil *s* d'une liste générée par la procédure *GenererListe* paramétrée avec la même valeur de seuil,
2. Ecrire le programme principal qui teste votre sous-programme en affichant les valeurs du tableau générer.

Question 3 – Exceptions

Compléter n'importe quelle version du programme de l'exercice 3 afin qu'il gère des exceptions que vous choisirez.