Examen – Informatique 2

Questions de cours : [4 points]

1) Dans les sous-programmes, quelle est la différence entre les paramètres formels et les paramètres effectifs ? **(1 point)**
2) Citer deux principales différences entre un type tableau à une dimension et un type enregistrement. **(1 point)**
3) Dans les enregistrements, quel est le rôle de l'instruction WITH ? **(1 point)**
4) Réécrire le corps du programme ci-contre en utilisant l'instruction WITH. **(1 point)**

```
Program Exercice1;
Type  Livre = Record
          Num: Integer;
          Auteur: String[20];
          Titre: String[50];
      End;
Var   N: Integer;   L: Livre;
Begin
  Read(N);
  Read(L.Num, L.Auteur, L.Titre);
End.
```

Exercice 01 : [8 points]

Partie A : (3 points)

Soient T un vecteur de N nombres entiers et X un nombre entier. Écrire un programme Pascal qui calcule et affiche le produit « PM » des composantes de T qui sont des multiples de X.

N.B : Nous considérons que les entrées N et X du programme sont strictement positives.

Exemple : Si on a : T | 15 | 7 | 9 | 2 | 6 | 4 | et X = 3 ⇨ PM = 15×9×6 = 810

Partie B : (5 points)

Soit A une matrice carrée de taille (N×N) de type réel. Écrire en Pascal un seul programme réalisant les opérations suivantes :

1) Rechercher le maximum « MAX » des composantes de A situées sur sa diagonale principale.
2) Calculer le nombre de composantes « NF » de la matrice A qui sont strictement inférieures à la valeur « MAX » et qui ne font pas parties de sa diagonale principale.
3) Afficher les résultats dans le programme.

Exercice 02 : [8 points]

Soit le programme Pascal suivant :

```
Program Exercice3;
 Var    X, Xi : Integer;
Function  Sous_Prog (N: Integer) : Integer;
  Var   Ni: Integer;
 Begin
        Ni := 0;
       While (N<> 0) Do
        Begin
          Ni :=  (Ni*10) + (N mod 10);
          N :=  N div 10;
        End;
   Sous_Prog := Ni;
 End;
BEGIN  {Début du programme principal}
   Read(X);
   Xi := Sous_Prog (X);
   Write(' Le résultat est : ', Xi);
END.  {Fin du programme principal}
```

Questions :

1) Quel est le type du résultat de la fonction ? **(0.25 point)**
2) Dérouler le programme pour X = 35. **(3.75 points)**
3) Déduire ce que fait le programme. **(0.5 point)**
4) Réécrire le programme en transformant la **fonction** en une **procédure** de même nom. **(2 points)**
5) Réécrire uniquement la fonction **Sous_Prog** afin que celle-ci vérifie si un nombre entier N est **palindrome ou non**. **(1.5 point)**

   Remarque **:** Un nombre est dit **palindrome** si la séquence de ses chiffres est la même lorsqu'elle est lue de droite à gauche ou de gauche à droite.

   Exemples **:** 2, 44, 606, 8118

Bonne chance

Corrigé d'examen – Informatique 2

Solution des questions de cours : [4 points]

1) **Dans les sous-programmes, quelle est la différence entre les paramètres formels et les paramètres effectifs ? (1 point)**

Réponse : (On note précisément les mots soulignés)

Les paramètres **formels** sont ceux **définis lors de la déclaration du sous-programme (ou définis dans l'entête du sous-programme).** **(0.5 point)**

Les paramètres **effectifs** sont ceux **utilisés lors de l'appel au sous-programme (0.5 point).**

2) **Citer les deux principales différences entre un type tableau à une dimension et un type enregistrement. (1 point)**

Critères de différence \ Type	Tableau	Enregistrement
Type de données	De **même type** **(0.25 point)**	Les données peuvent être de **types différents** **(0.25 point)**
Accès aux données	Par **indice** **(0.25 point)**	Par **identificateur du champ (ou attribut)** **(0.25 point)**

Remarques :

- La forme de la réponse (sous forme tableau ou texte) ne compte pas. On note seulement les mots soulignés.
- Les réponses liées à la syntaxe ne seront pas acceptées, par exemple : tableau=Array et enregistrement=Record.
- La différence en terme du nombre de composantes de chaque type peut être aussi acceptée. Un tableau peut avoir un nombre important (voire illimité) de composantes, contrairement à un enregistrement qui doit avoir un nombre limité et prédéfini de composantes.

3) **Dans les enregistrements, quel est le rôle de l'instruction WITH ? (1 point)**

L'instruction WITH permet **d'éviter les répétions de la variable enregistrement (0.5 point) lors de l'accès à ses différents champs. (0.5 point)**

Ou :

L'instruction WITH permet **d'éviter les répétitions de la notation pointée**
(Id_Var_Enreg. Id_Champ) (0.5 point) lors de l'accès à ses différents champs. (0.5 point)

Ou encore :

L'instruction WITH permet **d'utiliser les champs de l'enregistrement (0.5 point) sans répéter l'écriture de la variable d'enregistrement. (0.5 point)**

4) Réécrire le corps du programme ci-contre en utilisant l'instruction WITH. (1 point)

```
Begin
  Read (N);                          (0.25 point) //Exclusion de read(N) du bloc With

    With L do                        (0.25 point) //Syntaxe de With
      Begin
        Read (Num, Auteur, Titre);   (0.25 point) //0.25 point pour tous les champs sans l'écriture de « L. »
      End;                           (0.25 point) //Ajout des deux mots clés Begin et End;
End.
```

Ou bien :

```
Begin
  Read (N);                          (0.25 point) //Exclusion de read(N) du bloc With

    With L do                        (0.25 point) //Syntaxe de With
      Read (Num, Auteur, Titre);     (0.5 point) //0.5 point pour tous les champs sans l'écriture de « L. »
End.
```

Solution de l'exercice 01 : [8 points]

Partie A : (3 points)

Soient T un vecteur de N nombres entiers et X un nombre entier. Écrire un programme Pascal qui calcule et affiche le produit « PM » des composantes de T qui sont des multiples de X.

N.B : Nous considérons que les entrées N et X du programme sont strictement positives.

Programme pascal

```
Program exercice_2A;
Var
T: array [1 .. 20] of integer;      (0.25 point)
N, i, X, PM: integer;               (0.25 point)

Begin
{*-*-* Les entrées *-*-*}

  Write ('Introduire les valeurs de N et X :') ;

  Read (N, X);          (0.25 point)

Writeln ('Introduire les ',N,' composantes du vecteur T :') ;
For i:= 1 to N do       (0.25 point)
    Read (T[i]);        (0.25 point)

{*-*-* Le traitement *-*-*}

PM :=1;           (0.25 point)

For i:= 1 to N do             (0.25 point)
   If (T[i] mod X =0) Then    (0.75 point : 0.25 point pour la syntaxe de IF et 0.5 pour la condition)
       PM := PM*T[i];         (0.25 point)

{*-*-* Les sorties *-*-*}
Write ('Le produit des multiples de ',X, ' = ', PM) ;          (0.25 point)
End.
```

Partie B : (5 points)

Soit A une matrice carrée de taille (N×N) de type réel. Écrire en Pascal un seul programme réalisant les opérations suivantes :

1) Rechercher le maximum « MAX » des composantes de A situées sur sa diagonale principale.
2) Calculer le nombre de composantes « NF » de (la matrice) A qui sont strictement inférieures à la valeur « MAX » et qui ne font pas parties de sa diagonale principale.
3) Afficher les résultats dans le programme.

Programme pascal

```
Program exercice_2B;
  Var
    A : array [1..100, 1..100] of  real;    (0.25 point)
    i, j, N, NF: integer;                   (0.25 point)
    MAX : real ;                            (0.25 point)
  Begin

    {*-*-* Les entrées *-*-*}
   Write('Introduire le nombre de lignes ou de colonnes de A :') ;

    Read(N);          (0.25 point)
   Writeln('Introduire les ', n*n,' composantes de A :') ;
    For i:= 1 to N do
      For j:= 1 to N do                     (0.25 point)
        Read(A[i, j]);

    {*-*-* Traitement *-*-*}

MAX := A [1, 1] ;  <-------- (0.25 point)

For i:= 2 to N do       (0.25 point)
   If (A [i , i] > MAX) Then        (0.5 point : 0.25 point pour la syntaxe de IF et 0.25 pour la condition)
        MAX := A [i , i];  <-------- (0.25 point)

NF:=0;          (0.25 point)
For i:= 1 to N do          (0.25 point)
  For j:= 1 to N do          (0.25 point)

     if (i <> j) and (A [i , j] < MAX) then  <-------- (1.25 point : 0.25 point pour la syntaxe de IF et 0.5 pour chacune des 2 conditions)

        NF := NF + 1;  <-------- (0.25 point)

    {*-*-* Les sorties *-*-*}

     Writeln('MAX =', MAX:0:2) ;
     Write ('Nombre de valeurs < MAX = ', NF) ;     (0.25 point)
End.
```

Solution de l'exercice 02 : [8 points]

1) **Quel est le type du résultat de la fonction ? (0.25 point)**

   Le type est **Entier (ou Integer)** (0.25 point)

2) **Dérouler le programme pour X = 35. (3.75 points)**

   Nous allons utiliser dans ce qui suit les acronymes suivants :

   SP : Sous-Programme PP : Programme Principal

Instructions	Variables du PP		Variables du SP			Affichage
	X	Xi	N	Ni	Sous_Prog	
Read(X)	**35**	/	/	/	/	**0.25**
Xi := Sous_Prog(X) Xi := Sous_Prog(35) Appel à la fonction Sous_Prog Et transmission ou passage du paramètre X par valeur	**35**	**Xi := ?**	**35**	/	/	**0.25** **0.25** **//Transmission du paramètre au SP**
Ni := 0				**0**	/	**0.25**
While (N <> 0) **do** **(35 <> 0) True** Ni := (Ni*10) + (N mod 10) Ni := (0*10) + (35 mod 10) Ni := 0 + (35 mod 10) Ni := 0 + 5 **Ni := 5** N := N div 10 N := 35 div 10 **N := 3**			**3**	**5**	/	**0.25** **0.25** **0.25**
While (N <> 0) **do** **(3 <> 0) True** Ni := (Ni*10) + (N mod 10) Ni := (5*10) + (3 mod 10) Ni := 50 + (3 mod 10) Ni := 50 + 3 **Ni := 53** N := N div 10 N := 3 div 10 **N := 0**			**0**	**53**	/	**0.25** **0.25** **0.25**
While (N <> 0) **do** **(0 <> 0) False** **La condition n'est pas vérifiée, on sort de la boucle**					**/**	**0.25**
Sous_Prog := Ni **Sous_Prog := 53** Le résultat est retourné au PP		**53**			**53**	**0.25** **0.25 // Retour du Résultat dans PP**
Write(' Le résultat est : ', Xi);	**35**	**53**	**0**	**53**	**53**	Le résultat est : 53

0.25

0.25

Remarque : Dans la mise à jour de la valeur de Ni, l'évaluation de l'expression arithmétique n'est pas comptabilisée. Par exemple :

Ni := (Ni*10) + (N mod 10) → Ni := (0*10) + (35 mod 10) → Ni := 5 → La note 0.25 sera attribuée

3) Déduire ce que fait le programme. (0.5 point)

Le programme calcule et affiche l'inverse des chiffres (ou l'inverse de la séquence des chiffres) d'un nombre X (ou d'un nombre entier X) **(0.25 point)**, en utilisant une fonction (ou un sous-programme ou encore Sous_Prog). **(0.25 point)**

4) Réécrire le programme en transformant la *fonction* en une *procédure* de même nom. (2 points)

Programme Pascal

```
Program Exercice3;
Var   X, Xi : Integer;
Procedure  Sous_Prog (N: Integer; Var  Ni: Integer) ; : Integer;
  0.25                                0.25   0.25        0.25  //Suppression du type de la fonction

 Var  Ni: Integer;  0.25   //Suppression de la variable locale contenant le résultat de la fonction
                           //Suppression du mot clé « var » n'est pas comptabilisée
 Begin
        Ni := 0;
       While (N<> 0) Do
        Begin
            Ni :=  (Ni*10) + (N mod 10);
            N :=  N div 10;
        End;
  Sous_Prog := Ni;   0.25          //Suppression de la dernière instruction de la fonction
 End;

Begin  {Début du programme principal}
    Read(X);
    Sous_Prog (X, Xi);   0.5                  //Modification de l'appel au SP
    Write(' Le résultat est : ', Xi);
End.  {Fin du programme principal}
```

5) Réécrire uniquement la fonction *Sous_Prog* afin que celle-ci vérifie si un nombre entier N est palindrome ou non. (1.5 point)

Deux solutions sont possibles, nous optons, soit pour **le type de fonction booléen,** soit pour **le type de fonction entier**.

La fonction Sous_Prog « Solution 1 »

```
Function  Sous_Prog (N: Integer) : Boolean;
                                   0.25
 Var   Ni: Integer;
         S: Integer;   0.25      // Déclaration d'une variable locale de type entier
 Begin
        Ni := 0;
        S := N;  0.25              // Sauvegarde de la valeur de N dans la nouvelle variable
       While (N<> 0) Do
        Begin
          Ni :=  (Ni*10) + (N mod 10);
          N :=  N div 10;
        End;
     If (S = Ni) then  0.25   // Condition + Syntaxe de If
       Sous_Prog : = True  0.25
    Else
       Sous_Prog := False;  0.25
End;
```

La fonction Sous_Prog « Solution 2 »

```
Function  Sous_Prog (N: Integer) : Integer;
                                   0.25
 Var   Ni: Integer;
         S: Integer;   0.25      // Déclaration d'une variable locale de type entier
 Begin
        Ni := 0;
        S := N;  0.25              // Sauvegarde de la valeur de N dans la nouvelle variable
       While (N<> 0) Do
        Begin
          Ni :=  (Ni*10) + (N mod 10);
          N :=  N div 10;
        End;
     If (S = Ni) then  0.25  // Condition + Syntaxe de If
       Sous_Prog : = 1  0.25
    Else
       Sous_Prog := 0;  0.25
End;
```