Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE

Ministère du Travail, des Relations sociales, de la Famille, de la Solidarité et de la Ville
Ministère de la Santé et des Sports
Haut Commissaire à la Jeunesse

EXAMEN PROFESSIONNEL POUR LA VERIFICATION D'APTITUDE AUX FONCTIONS DE CHEF PROGRAMMEUR

ANNEE 2009

JEUDI 1er OCTOBRE 2009
13 h 00 à 18 h 00
(horaire métropole)

EPREUVE – durée : 5 heures – coefficient : 5

IMPORTANT :
Les candidats sont priés de vérifier le nombre de pages et la numérotation des documents joints.

Vous devez choisir un langage et un seul, parmi ceux proposés, dans lequel vous composerez votre devoir et l'indiquerez au début de votre copie.

Vous devez répondre aux seuls exercices associés à ce langage.

Sans le respect de ces deux règles votre copie ne pourra pas être prise en compte.

SUJET A TRAITER

Analyse critique d'un programme rédigé dans un langage évolué choisi par le candidat

— Informix 4 GL ..... page 2 à 7

— C++ ..... page 8 à 14

— C# ..... page 15 à 21

— Java ..... page 22 à 28

— PHP ..... page 29 à 35

— XSLT ..... page 36 à 46

— Visual Basic 6 ..... Page 47 à 53

Recommandations :

Vous devez choisir un langage et un seul, parmi ceux proposés, dans lequel vous composerez obligatoirement l'intégralité de votre devoir et vous indiquerez le langage choisi au début de votre copie. **Si cette règle n'est pas respectée, votre copie ne sera pas prise en compte.**

Toutes les réponses doivent figurer dans la copie (ne pas écrire directement sur le sujet). Pour les parties relatives aux corrections de code vous voudrez bien indiquer le numéro des lignes à ajouter (ligne précédente suivie de a, b, c ...), à modifier ou à supprimer de la manière suivante :

7a *instruction à jouter*
7b *instruction à ajouter*
Supprimer ligne 8
9 *instruction modifiée*

INFORMIX 4GL

1 Questions (4 points)

1.1 QUESTION N° 1

Citer tous les types d'itérations possibles associés à ce langage et justifier l'intérêt de choisir chaque type. Donner les syntaxes correspondantes.

1.2 QUESTION N° 2

Qu'est-ce qu'une fonction récursive ?

Dans quels cas l'utiliser ?

Pourquoi l'utilisation de telles fonctions n'est elle pas toujours recommandée ? (expliciter votre réponse)

Qu'est-ce qu'une fonction récursive terminale ?

2 Exercices (8 points)

2.1 EXERCICE N° 1

```
1. FUNCTION GetTotal(v_sale1, v_sale2, v_sale3, v_remise)
2. DEFINE v_sale1 LIKE sale.sale1
3. DEFINE  v_sale2 LIKE sale.sale2
4. DEFINE v_sale3 LIKE sale.sale3
5. DEFINE v_remise LIKE sale.remise
6. DEFINE v_total decimal(10,2)
7. LET v_total = GetTotal(v_sale1, v_sale2 , v_sale3)
```

```
8.  call salestax(v_total, v_remise)
9.  END FUNCTION
10. FUNCTION AddSales(v_sale1, v_sale2 , v_sale3)
11. LET v_total = v_sale1 + v_sale2 + v_sale3
12. return v_total
13. END FUNCTION
14. FUNCTION salestax(v_total, v_remise)
15. DEFINE v_total decimal(10,2)
16. DEFINE v_remise decimal(6,3)
17. DEFINE SalesTax decimal(10,2)
18. LET SalesTax = v_total * .055
19. LET SalesTax = SalesTax - (v_remise * v_total)
20. display "Taxation totale :  ",SalesTax
21. END FUNCTION
```

2.1.1 Quel est le but de ce code ?

2.1.2 Décrire la structure de ce code

2.1.3 Décrire sommairement les fonctions présentes dans ce code

2.1.4 Corriger ce code afin que les procédures s'exécutent correctement

2.1.5 Critiquer ce code et proposer des améliorations (sans réécrire le code)

2.2 EXERCICE N° 2

```
1.  FUNCTION Test_Pwd(v_login, v_passwd)
2.  define v_login like user.login
3.  define v_passwd like user.passwd
4.  define loginsucceded smallint
5.  define msg char(50)
6.  if v_passwd = = « password »
7.  then
8.  let loginsucceded = true
9.  else
10. let loginsucceded = false
11. let msg = « \nMot de passé  invalide, réessayez !\n »
12. call Msg_Output(msg)vi
13. end if
14. END FUNCTION
15. FUNCTION Msg_Output(vdisplay)
16. define vdisplay char(512)
17. display vdisplay clipped
18. END FUNCTION
```

2.2.1 Quel est le but de ce code ?

2.2.2 Donner la signification de l'instruction suivante : **clipped**

2.2.3 Ajouter les instructions nécessaires à la bonne exécution de ce code :

2.2.4 Critiquer ce code ainsi corrigé et proposer des améliorations (sans réécrire le code)

2.3 EXERCICE N° 3

```
1. FUNCTION  str_add(si,ll,car)
2. define si char(80)
3. define ll smallint
4. define car char(1)
5. define k,l smallint
6. define s char(80)
7. if si is NULL or length(si) = 0
8. then
9. return ""
10. end if
11. let s = car
12. for k=2 to ll
13. let s = s[1,k-1], car
14. end for
15. let l=length(si)
16. if l = ll
17. then
18. return si
19. end if
20. let s = s[1,ll-l], si clipped
21. return s
22. END FUNCTION
```

2.3.1 Quel est le but de ce code ?

2.3.2 Quel serait le résultat produit par l'instruction suivante :
Str_add (« 321 », 5, « 0 »)

2.4 EXERCICE N° 4

L'extrait de code 4gl présenté ci-dessous permet de créer une transaction

```
1. FUNCTION B_Work()
2. whenever error continue
3. if inwork=0
4. then
5. BEGIN WORK
6. if status != 0
7. then
8. let msg = "Probleme sur BEGIN WORK : ",status,"\n"
9. call Msg_Output(msg)
```

```
10. return FALSE
11. end if
12. let inwork=1
13. end if
14. whenever error stop
15. return TRUE
16. END FUNCTION
```

2.4.1 Donner la signification de la variable suivante :

inwork

2.4.2 Créer la fonction complémentaire, permettant d'abandonner la transaction sans effectuer de mise à jour dans la base de données

2.4.3 Expliquez l'intérêt de créer une transaction.

2.5 EXERCICE N° 5

```
1. FUNCTION Fct_U2(v_e)
2. define v_e integer
3. if v_e > 0
4. let v_e = v_e – 1
5. return 3 * Fct_U2(v_e) + 1
6. else
7. return 1
8. end if
9. END FUNCTION
```

2.5.1 Quel est le but de ce code ?

2.5.2 Corriger ce code.

2.5.3 Réécrivez cette fonction sans faire appel à la récursivité.

3 Problème (8 points)

Il sera tenu compte de la clarté de la rédaction du code 4GL
Vous prendrez toute initiative que vous jugerez nécessaire afin de pouvoir réaliser le sujet.
Toutefois, vous devrez les faire apparaître clairement dans votre devoir.

```
1. FUNCTION AjoutEnr(v_nodisque, v_prix, v_editeur, v_genre, v_nom, v_prenom, v_tit
2. re, v_nbexemplaire)
3. DEFINE v_nodisque LIKE disque.nodisque
4. DEFINE v_prix LIKE disque.prix
5. DEFINE v_editeur LIKE disque.editeur
6. DEFINE v_genre LIKE disque.genre
7. DEFINE v_nom LIKE disque.nom
8. DEFINE v_prenom LIKE disque.prenom
```

```
9.  DEFINE v_titre LIKE disque.titre
10. DEFINE v_nbexemplaire LIKE disque.nbexemplaire
11. IF v_nodisque!="" AND v_prix > 0
12. THEN
13. LET v_nom=UPSHIFT(v_nom)
14. INSERT INTO disque (nodisque, prix, editeur, genre, nom, prenom, titre, nbexempl
15. aire)
16. VALUES (v_noDisque, v_prix, v_editeur, v_genre, v_nom, v_prenom, v_titre, v_nbex
17. emplaire)
18. END IF
19. IF status < 0
20. let msg = "Probleme sur INSERT : ",status,"\n"
21. call Msg_Output(msg)
22. return false
23. END IF
24. return true
25. END FUNCTION
26. FUNCTION ListeEnreg()
27. DEFINE v_genre LIKE disque.genre
28. DEFINE v_nom LIKE disque.nom
29. DEFINE v_titre LIKE disque.titre
30. DECLARE disq CURSOR FOR
31. SELECT genre, nom, titre FROM disque WHERE nom="JACKSON" ORDER BY 1, 2
32. FOREACH disq INTO v_genre
33. DISPLAY v_genre
34. DISPLAY v_nom
35. DISPLAY v_titre
36. END FOREACH
37. END FUNCTION
38. FUNCTION NbEnreg()
39. DEFINE v_nbenreg integer
40. DECLARE disq CURSOR FOR
41. SELECT count(*) FROM disque
42. FOREACH disq INTO v_nbenreg
43. END FOREACH
44. return v_nbenreg
45. END FUNCTION
46. FUNCTION NbAlbumJackson()
47. DEFINE v_nbenreg integer
48. DECLARE disq CURSOR FOR
49. SELECT count(*) FROM disque WHERE nom="JACKSON"
50. FOREACH disq INTO v_nbenreg
51. END FOREACH
52. return v_nbenreg
53. END FUNCTION
54. FUNCTION PrixMoyen()
```

```
55. DEFINE v_PrixTotal decimal (10,2)
56. DEFINE cpt integer
57. DEFINE v_prix LIKE disque.prix
58. LET v_PrixTotal=0
59. LET cpt=0
60. DECLARE disq CURSOR FOR
61. SELECT prix FROM disque
62. FOREACH disq INTO v_prix
63. LET v_PrixTotal=v_PrixTotal+v_prix
64. LET cpt=cpt+1
65. END FOREACH
66. return v_PrixTotal/cpt
67. END FUNCTION
```

3.1.1 Quel est l'objectif de cet extrait de programme ?

3.1.2 Corriger ces fonctions afin qu'elles s'exécutent correctement

3.1.3 Critiquer ce code et proposer des améliorations (sans réécrire le code)

3.1.4 Ecrire les fonctions qui permettent :

- De supprimer un enregistrement
- De se déplacer vers l'enregistrement suivant, lors de l'appui sur la commande « Suivant »
- De calculer la proportion de disques (en %) dont le nom est « JACKSON », de stocker le résultat dans une variable et d'afficher le résultat
- De calculer le coût des exemplaires de l'album dont le titre est « Thriller ».

Recommandations :

Vous devez choisir un langage et un seul, parmi ceux proposés, dans lequel vous composerez obligatoirement l'intégralité de votre devoir et vous indiquerez le langage choisi au début de votre copie. **Si cette règle n'est pas respectée, votre copie ne sera pas prise en compte.**

Toutes les réponses doivent figurer dans la copie (ne pas écrire directement sur le sujet). Pour les parties relatives aux corrections de code vous voudrez bien indiquer le numéro des lignes à ajouter (ligne précédente suivie de a, b, c ...), à modifier ou à supprimer de la manière suivante :

7a *instruction à jouter*
7b *instruction à ajauter*
Supprimer ligne 8
9 *instruction modifiée*

C++

1 Questions (sur 4 points)

1.1 Question n° 1

Citez tous les types d'itération possibles associés à ce langage et justifiez l'intérêt de choisir chaque type, donner les syntaxes correspondantes.

1.2 Question n° 2

Qu'est-ce qu'une fonction récursive ?

Dans quels cas l'utiliser ?

Pourquoi n'est-il pas toujours conseillé d'utiliser de telles fonctions ? (expliciter votre réponse)

Qu'est-ce qu'une fonction récursive terminale ?

2 Exercices (sur 8 points)

2.1 Exercice n° 1

```
1. public static boolean isSet()
2. float VarVente1, VarVente2, Varvente3, VarRemise,
3. if(isset(Var_POST["fVente1"]) && Var_POST["fVente1"]!="")
4. {
5. VarVente1=Var_POST["fVente1"];
6. VarVente2=Var_POST["f Vente2"];
7. VarVente3=Var_POST["f Vente3"];
8. VarRemise=Var_POST["fRemise"];
```

```
9.  }
10. VarTotal = GetTotal(Vente1,Var Vente2, Var Vente3);
11. float VentesTaxe(VarTotal, VarRemise);
12. float GetTotal(VarVente1, VarVente2, VarVente3) {
13. float VarTotal ;
14. VarTotal = Var Vente1 + Var Vente2 + Var Vente3;
15. return VarTotal ;
16. }
17. float  VentesTaxe(VarTotal, VarRemise) {
18. float VarVentesTaxes;
19. VarVentesTaxe = (VarTotal * .055)
20. VarVentesTaxe = VarVentesTaxe - (VarRemise * VarTotal)
21. System.out.println( "Taxation totale :  ".VarVentesTaxe)
22. }
```

2.1.1 Quel est le but de ce code ?

2.1.2 Décrire la structure de ce code

2.1.3 Décrire sommairement les instructions présentes dans ce code

2.1.4 Corriger ce code afin que les procédures s'exécutent correctement

2.1.5 Critiquer ce code et proposer des améliorations (sans réécrire le code)

2.2 EXERCICE N° 2

```
1.  public static boolean isSet()
2.  if(isset(Var_POST["fPassword"]) && Var_POST["fPassword"]!="")
3.  {
4.  VarLogin=Var_POST["fLogin"];
5.  VarPassword=Var_POST["fPassword"];
6.  if (VarPassword=="password") {
7.  Var_SESSION["LoginSucceeded"]=True ;
8.  Var_SESSION["Login"]=VarLogin ;
9.  Var_SESSION["Password"]=VarPassword ;
10. }
11. else {
12. Var_SESSION["LoginSucceeded"]=False ;
13. Var_SESSION["Login"]=VarLogin ;
14. Var_SESSION["Password"]=VarPassword ;
15. System.out.print "Mot de passe non valide, r&eacute;essayez !";        //
16. }
17. ;
```

2.2.1 Quel est le but de ce code ?

2.2.2 A quoi sert l'instruction suivante :

Var_SESSION["LoginSucceeded"]=True ;

2.2.3 Ajouter le code nécessaire à la bonne exécution de ce code :

2.2.4 Critiquer le code ainsi corrigé et proposer des améliorations (sans réécrire le code)

2.3 EXERCICE N° 3

```
1. public class RegexpWrapper
2. {
3. RegexpWrapper()
4. {
5. };
6. public string patt, subj
7. public static boolean ereg(patt, subj)
8. {
9. Pattern p = Pattern.compile(patt);
10. Matcher m = p.matcher(subj);
11. public boolean b=m.matches();
12. return b;
13. }
14. }
15. • } public boolean isMultiline () *****Function IsMultiLine(VarBuffer, VarUnixConvention) {
16. VarLineSeparator=chr(10).chr(13);
17. If (VarUnixConvention) {
18. VarLineSeparator=chr(10);
19. }
20. if(ereg(VarLineSeparator,VarBuffer)){
21. return true;
22. }
```

2.3.1 **IsMultiline** : Quel est le but de ce code ?

2.3.2 A quoi sert la fonction suivante :

ereg (string patt , string subj [, array &Varregs])

2.4 EXERCICE N° 4

L'EXTRAIT DU CODE PRESENTE CI-DESSOUS PERMET DE COPIER UN FICHIER SOURCE VERS UNE DESTINATION

```
public class IO
{
public static void main (String File[])
{
...
File file1 = new File("C:/tmo/myfile1.txt");
File file2 = new File("C:/tmo/myfile2.txt");
this.myFileCopy(file1,file2);
}
private boolean myFileCopy(File src,File dst) throws IOException
        {
```

1. public static string MyFileCopy(string Varsource, string Vardestination, string VarOverwrite)
   {
2. Varfichierouvert = fopen(Varsource, "r");
3. Varcontenu ="";
4. while (!feof(Varfichierouvert)) {
5. Varcontenu .= fread(Varfichierouvert, 8192);
6. }
7. fclose (Varfichierouvert);
8. if (VarOverwrite==true) {
9. Varfichierouvert = fopen(Vardestination, "w+");
10. }else {
11. Varfichierouvert = fopen(Vardestination, "a+"};
12. }
13. if (!fwrite(Varfichierouvert, Varcontenu)) {
14. echo "Impossible d'écrire dans le fichier". Varfilename;
15. exit;
16. }
17. fclose (Varfichierouvert);
18. }

2.4.1 Donner la signification de la variable suivante :

VarOverwrite

2.4.2 Ecrivez le code faisant appel à la fonction MyFileCopy en écrivant 2 appels possibles pour un même nom de fichier et commenter ce code.

2.4.3 Expliquez la transmission des variables par référence ou par valeur.

2.5 EXERCICE N° 5

```
1. public int Vare
2. public static FctU2(Vare){
3. if (Vare>0) {
4. Vare-1;
5. return 3*FctU2(Vare)+1;
6. } else  {
7. return 1;
8. )
9. )
```

2.5.1 Quel est le but de ce code ?

2.5.2 Corriger ce code.

2.5.3 Réécrivez cette fonction sans faire appel à la récursivité.

3 Problème (sur 8 points)

Il sera tenu compte de la clarté de votre rédaction du code C++.
Vous prendrez toute initiative que vous jugerez nécessaire afin de pouvoir réaliser le sujet.
Toutefois, vous devrez les faire apparaître clairement dans votre devoir.

```
1. ...
2. fTabDisques=AjoutEnreg(fTabDisques, NoDisque, Prix, Editeur, Genre, Nom, Prenom, Titre,
   NbExemplaire);
3. echo ListeEnreg(fTabDisques);
4. echo NbEnreg(fTabDisques)."</br>";
5. echo NbAlbumJackson(fTabDisques)."</br>";
6. echo PrixMoyen(fTabDisques)."</br>";
7.  public static  AjoutEnreg(fTabDisques, NoDisque, Prix, Editeur, Genre, Nom, Prenom, Titre,
   NbExemplaire){
8. if (NoDisque!=""&&Prix>0){
9.  fLigneDisque["dis_id"]=NoDisque;
10. fLigneDisque["dis_prix"]=round(Prix, "2");
11. fLigneDisque["dis_edit"]=Editeur;
12. fLigneDisque["dis_genre"]=Genre;
13. fLigneDisque["dis_nom"]=strtoupper(Nom);
14. fLigneDisque["dis_prenom"]=Prenom;
15. fLigneDisque["dis_titre"]=Titre;
16. fLigneDisque["dis_nbex"]=NbExemplaire;
17. fTabDisques[]=fLigneDisque;
```

```
18. }
19. return fTabDisques;
20. }
21. public static ListeEnreg(fTabDisques) {
22. // Trier la table
23. foreach (fTabDisques as fLigneDisque) {
24. if (fLigneDisque["dis_nom"]=="JACKSON") {
25. echo fLigneDisque["dis_genre"]."\t";
26. echo fLigneDisque["dis_nom"]."\t";
27. echo fLigneDisque["dis_titre"]."</br>";
28. }
29. }
30. }
31. public static NbEnreg() {
32. NbEnreg=sizeof(fTabDisques);
33. return NbEnreg;
34. }
35. public static NbAlbumJackson(fTabDisques) {
36. foreach (fTabDisques as fLigneDisque) {
37. if (fLigneDisque["dis_nom"}="JACKSON"} {
38. NbEnreg++;
39. }
40. }
41. return NbEnreg
42. }
43. public static PrixMoyen(fTabDisques) {
44. PrixTotal=0;
45. i=0;
46. foreach (fTabDisques as fLigneDisque) {
47. i++
48. PrixTotal=PrixTotal+fLigneDisque["dis_prix"];
49. }
50. return PrixTotal/i;
51. }
```

3.1.1 Quel est l'objectif de cet extrait de programme ?

3.1.2 Corriger ce code afin qu'il s'exécute correctement

3.1.3 Critiquer ce code et proposer des améliorations (sans réécrire le code)

3.1.4 Ecrire les parties de programme qui permettent :

- De supprimer un enregistrement après avoir demandé confirmation
- De se déplacer vers l'enregistrement suivant, lors de l'appui sur la commande « Suivant »
- De calculer la proportion de disques (en %) dont le nom est « JACKSON », de stocker le résultat dans une variable et d'afficher le résultat
- De calculer le coût total des exemplaires de l'album dont le titre est « Thriller ».

Recommandations :

Vous devez choisir un langage et un seul, parmi ceux proposés, dans lequel vous composerez obligatoirement l'intégralité de votre devoir et vous indiquerez le langage choisi au début de votre copie. **Si cette règle n'est pas respectée, votre copie ne sera pas prise en compte.**

Toutes les réponses doivent figurer dans la copie (ne pas écrire directement sur le sujet). Pour les parties relatives aux corrections de code vous voudrez bien indiquer le numéro des lignes à ajouter (ligne précédente suivie de a, b, c ...), à modifier ou à supprimer de la manière suivante :

7a *instruction à jouter*
7b *instruction à ajouter*
Supprimer ligne 8
9 *instruction modifiée*

C#

1 Questions (sur 4 points)

1.1 Question n° 1

Citez tous les types d'itération possibles associés à ce langage et justifiez l'intérêt de choisir chaque type.

1.2 Question n° 2

Qu'est-ce qu'une fonction récursive ?

Dans quels cas l'utiliser ?

Pourquoi n'est-il pas toujours conseillé d'utiliser de telles fonctions ? (expliciter votre réponse)

Qu'est-ce qu'une fonction récursive terminale ?

2 Exercices (sur 8 points)

2.1 Exercice n° 1

```
1. public static boolean isSet()
2. float VarVente1, VarVente2, Varvente3, VarRemise,
3. if(isset(Var_POST["fVente1"]) && Var_POST["fVente1"]!="")
4. {
5. VarVente1=Var_POST["fVente1"];
6. VarVente2=Var_POST["f Vente2"];
7. VarVente3=Var_POST["f Vente3"];
```

```
8.  VarRemise=Var_POST["fRemise"];
9.  }
10. VarTotal = GetTotal(Vente1,Var Vente2, Var Vente3);
11. float VentesTaxe(VarTotal, VarRemise);
12. float GetTotal(VarVente1, VarVente2, VarVente3) {
13. float VarTotal ;
14. VarTotal = Var Vente1 + Var Vente2 + Var Vente3;
15. return VarTotal ;
16. }
17. float  VentesTaxe(VarTotal, VarRemise) {
18. float VarVentesTaxes;
19. VarVentesTaxe = (VarTotal * .055}
20. VarVentesTaxe = VarVentesTaxe - (VarRemise * VarTotal)
21. System.out.print( "Taxation totale :  ".VarVentesTaxe}
22. }
23. ;
```

2.1.1 Quel est le but de ce code ?

2.1.2 Décrire la structure de ce code

2.1.3 Décrire sommairement les instructions présentes dans ce code

2.1.4 Corriger ce code afin que les procédures s'exécutent correctement

2.1.5 Critiquer ce code et proposer des améliorations (sans réécrire le code)

2.2 EXERCICE N° 2

```
1.  public static boolean isSet()
2.  if(isset(Var_POST["fPassword"]) && Var_POST["fPassword"]!="")
3.  {
4.  VarLogin=Var_POST["fLogin"];
5.  VarPassword=Var_POST["fPassword"];
6.  if (VarPassword=="password") {
7.  Var_SESSION["LoginSucceeded"]=True ;
8.  Var_SESSION["Login"]=VarLogin ;
9.  Var_SESSION["Password"]=VarPassword ;
10. }
11. else {
12. Var_SESSION["LoginSucceeded"]=False ;
13. Var_SESSION["Login"]=VarLogin ;
14. Var_SESSION["Password"]=VarPassword
15. System.out.print "Mot de passe non valide, r&eacute;essayez !";
16. }
17. ;
```

2.2.1 Quel est le but de ce code ?

2.2.2 A quoi sert l'instruction suivante :

Var_SESSION["LoginSucceeded"]=True ;

2.2.3 Ajouter le code nécessaire à la bonne exécution de ce code :

2.2.4 Critiquer ce code ainsi corrigé et proposer des améliorations (sans réécrire le code)

2.3 EXERCICE N° 3

```
1. public class RegexpWrapper
2.
3. RegexpWrapper()
4. {
5. };
6. public string patt, subj
7. public static boolean ereg(patt, subj)
8. {
9. Pattern p = Pattern.compile(patt);
10. Matcher m = p.matcher(subj);
11. public boolean b=m.matches();
12.  return b;
13. }
14. }
15. • } public boolean isMultiline () *****Function IsMultiLine(VarBuffer, VarUnixConvention) {
16. VarLineSeparator=chr(10).chr(13);
17. If (VarUnixConvention) {
18. VarLineSeparator=chr(10);
19. }
20. if(ereg(VarLineSeparator,VarBuffer)){
21. return true;
22. }
```

2.3.1 **IsMultiline** : Quel est le but de ce code ?

2.3.2 A quoi sert la fonction suivante :

ereg (string patt , string subj [, array &Varregs])

2.4 EXERCICE N° 4

L'EXTRAIT DU CODE PRESENTE CI-DESSOUS PERMET DE COPIER UN FICHIER SOURCE VERS UNE DESTINATION

```
public class IO
{
public static void main (String File[])
{
...
File file1 = new File("C:/tmo/myfile1.txt");
File file2 = new File("C:/tmo/myfile2.txt");
this.myFileCopy(file1,file2);
}
private boolean myFileCopy(File src,File dst) throws IOException
        {
```

1.
2. public static string MyFileCopy(string Varsource, string Vardestination, string VarOverwrite)
   {
3. Varfichierouvert = fopen(Varsource, "r");
4. Varcontenu ="";
5. while (!feof(Varfichierouvert)) {
6. Varcontenu .= fread(Varfichierouvert, 8192);
7. }
8. fclose (Varfichierouvert);
9. if (VarOverwrite==true) {
10. Varfichierouvert = fopen(Vardestination, "w+");
11. }else {
12. Varfichierouvert = fopen(Vardestination, "a+");
13. }
14. if (!fwrite(Varfichierouvert, Varcontenu)) {
15. echo "Impossible d'écrire dans le fichier". Varfilename;
16. exit;
17. }
18. fclose (Varfichierouvert);
19. }

2.4.1 Donner la signification de la variable suivante :

VarOverwrite

2.4.2 Ecrivez le code faisant appel à la fonction MyFileCopy en écrivant 2 appels possibles pour un même nom de fichier et commenter ce code.

2.4.3 Expliquez la transmission des variables par référence ou par valeur.

2.5 EXERCICE N° 5

```
1. public int Vare
2. public static FctU2(Vare){
3. if (Vare>0) {
4. Vare-1;
5. return 3*FctU2(Vare)+1;
6. } else  {
7. return 1;
8. }
9. }
```

2.5.1 Quel est le but de ce code ?

2.5.2 Corriger ce code.

2.5.3 Réécrivez cette fonction sans faire appel à la récursivité.

3 Problème (sur 8 points)

Il sera tenu compte de la clarté de la rédaction du code C#.
Vous prendrez toute initiative que vous jugerez nécessaire afin de pouvoir réaliser le sujet.
Toutefois, vous devrez les faire apparaître clairement dans votre devoir.

```
1. ...
2. fTabDisques=AjoutEnreg(fTabDisques, NoDisque, Prix, Editeur, Genre, Nom, Prenom, Titre,
   NbExemplaire);
3. echo ListeEnreg(fTabDisques);
4. echo NbEnreg(fTabDisques)."</br>";
5. echo NbAlbumJackson(fTabDisques)."</br>";
6. echo PrixMoyen(fTabDisques)."</br>";
7. public static  AjoutEnreg(fTabDisques, NoDisque, Prix, Editeur, Genre, Nom, Prenom, Titre,
   NbExemplaire){
8. if (NoDisque!=""&&Prix>0){
9.  fLigneDisque["dis_id"]=NoDisque;
10. fLigneDisque["dis_prix"]=round(Prix, "2");
11. fLigneDisque["dis_edit"]=Editeur;
12. fLigneDisque["dis_genre"]=Genre;
13. fLigneDisque["dis_nom"]=strtoupper(Nom);
14. fLigneDisque["dis_prenom"]=Prenom;
15. fLigneDisque["dis_titre"]=Titre;
16. fLigneDisque["dis_nbex"]=NbExemplaire;
```

```
17.  fTabDisques[]=fLigneDisque;
18. }
19. return fTabDisques;
20. }
21. public static ListeEnreg(fTabDisques) {
22. // Trier la table
23. foreach (fTabDisques as fLigneDisque) {
24. if (fLigneDisque["dis_nom"]=="JACKSON") {
25. echo fLigneDisque["dis_genre"]."\t";
26. echo fLigneDisque["dis_nom"]."\t";
27. echo fLigneDisque["dis_titre"]."</br>";
28. }
29. }
30. }
31. public static NbEnreg() {
32. NbEnreg=sizeof(fTabDisques);
33. return NbEnreg;
34. }
35. public static NbAlbumJackson(fTabDisques) {
36. foreach (fTabDisques as fLigneDisque) {
37. if (fLigneDisque["dis_nom"]="JACKSON") {
38. NbEnreg++;
39. }
40. }
41. return NbEnreg
42. }
43. public static PrixMoyen(fTabDisques) {
44. PrixTotal=0;
45. i=0;
46. foreach (fTabDisques as fLigneDisque) {
47. i++;
48. PrixTotal=PrixTotal+fLigneDisque["dis_prix"];
49. }
50. return PrixTotal/i;
51. }
```

3.1.1 Quel est l'objectif de cet extrait de programme ?

3.1.2 Corriger ce code afin qu'il s'exécute correctement

3.1.3 Critiquer ce code et proposer des améliorations (sans réécrire le code)

3.1.4 Ecrire les parties de programme qui permettent :

- De supprimer un enregistrement après avoir demandé confirmation
- De se déplacer vers l'enregistrement suivant, lors de l'appui sur la commande « Suivant »
- De calculer la proportion de disques (en %) dont le nom est « JACKSON », de stocker le résultat dans une variable et d'afficher le résultat
- De calculer le coût total des exemplaires de l'album dont le titre est « Thriller ».

Recommandations :

Vous devez choisir un langage et un seul, parmi ceux proposés, dans lequel vous composerez obligatoirement l'intégralité de votre devoir et vous indiquerez le langage choisi au début de votre copie. **Si cette règle n'est pas respectée, votre copie ne sera pas prise en compte.**

Toutes les réponses doivent figurer dans la copie (ne pas écrire directement sur le sujet). Pour les parties relatives aux corrections de code vous voudrez bien indiquer le numéro des lignes à ajouter (ligne précédente suivie de a, b, c ...), à modifier ou à supprimer de la manière suivante :

7a *instruction à jouter*
7b *instruction à ajouter*
Supprimer ligne 8
9 *instruction modifiée*

JAVA

1 Questions (sur 4 points)

1.1 Question n° 1

Citez tous les types d'itération possibles associés à ce langage et justifiez l'intérêt de choisir chaque type, donner les syntaxes correspondantes.

1.2 Question n° 2

Qu'est-ce qu'une fonction récursive ?

Dans quels cas l'utiliser ?

Pourquoi n'est-il pas toujours conseillé d'utiliser de telles fonctions ? (expliciter votre réponse)

Qu'est-ce qu'une fonction récursive terminale ?

2 Exercices (sur 8 points)

2.1 Exercice n° 1

```
1. public static boolean isSet(java.lang.String flagName)
2. if(isset(Var_POST["fVente1"]) && Var_POST["fVente1"]!="")
3. {
4. VVente1=Var_POST["fVente1"];
5. VVente2=Var_POST["f Vente2"];
6. Var Vente3=Var_POST["f Vente3"];
7. VarRemise=Var_POST["fRemise"];
```

```
8.  }
9.  VarTotal = GetTotal(Var Vente1,Var Vente2, Var Vente3);
10. VentesTaxe(VarTotal, VarRemise);
11. function GetTotal(VarVente1, VarVente2, VarVente3) {
12. VarTotal = Var Vente1 + Var Vente2 + Var Vente3;
13. return VarTotal ;
14. }
15. function VentesTaxe(VarTotal, VarRemise) {
16. VarVentesTaxe = (VarTotal * .055)
17. VarVentesTaxe = VarVentesTaxe - (VarRemise * VarTotal)
18. System.out.println( "Taxation totale :  ".VarVentesTaxe)
19. }
20. ;
```

2.1.1 Quel est le but de ce code ?

2.1.2 Décrire la structure de ce code

2.1.3 Décrire sommairement les instructions présentes dans ce code

2.1.4 Corriger ce code afin que les procédures s'exécutent correctement

2.1.5 Critiquer ce code et proposer des améliorations (sans réécrire le code)

2.2 EXERCICE N° 2

```
1.  public static boolean isSet(java.lang.String flagName)
2.  if(isset(Var_POST["fPassword"]) && Var_POST["fPassword"]!="")
3.  {
4.  VarLogin=Var_POST["fLogin"];
5.  VarPassword=Var_POST["fPassword"];
6.  if (VarPassword=="password") {
7.  Var_SESSION["LoginSucceeded"]=True ;
8.  Var_SESSION["Login"]=VarLogin ;
9.  Var_SESSION["Password"]=VarPassword ;
10. }
11. else {
12. Var_SESSION["LoginSucceeded"]=False ;
13. Var_SESSION["Login"]=VarLogin ;
14. Var_SESSION["Password"]=VarPassword ;
15. System.out.print "Mot de passe non valide, r&eacute;essayez !";
16. }
17. ;
```

2.2.1 Quel est le but de ce code ?

2.2.2 A quoi sert l'instruction suivante :

Var_SESSION["LoginSucceeded"]=True ;

2.2.3 Ajouter le code nécessaire à la bonne exécution de ce code :

2.2.4 Critiquer ce code ainsi corrigé et proposer des améliorations (sans réécrire le code)

2.3 EXERCICE N° 3

```
1. import java.lang.String;
1. import java.util.regex.*;
2. public class RegexpWrapper
3. {
4. RegexpWrapper()
5. {
6. }
7. public static boolean ereg(String patt,String subj)
8. {
9. Pattern p = Pattern.compile(patt);
10. Matcher m = p.matcher(subj);
11. boolean b=m.matches();
12.  return b;
13. }
14.  }
15. • } public boolean isMultiline () *****Function IsMultiLine(VarBuffer, VarUnixConvention) {
16. VarLineSeparator=chr(10).chr(13);
17. If (VarUnixConvention) {
18. VarLineSeparator=chr{10);
19. }
20. if(ereg(VarLineSeparator,VarBuffer)){
21. return true;
22. }
```

2.3.1 **IsMultiline** : Quel est le but de ce code ?

2.3.2 A quoi sert la fonction suivante :

ereg (string patt , string subj [, array &Varregs])

2.4 EXERCICE N° 4

L'EXTRAIT DU CODE PRESENTE CI-DESSOUS PERMET DE COPIER UN FICHIER SOURCE VERS UNE DESTINATION

```
public class IO
    {
    public static void main (String File[])
    {
    ...
    File file1 = new File("C:/tmo/myfile1.txt");
    File file2 = new File("C:/tmo/myfile2.txt");
    this.myFileCopy(file1,file2);
    }
    private boolean myFileCopy(File src,File dst) throws IOException
        {
```

```
1.  public static string MyFileCopy( string Varsource, string Vardestination, string  VarOverwrite)
    {
2.  Varfichierouvert = fopen(Varsource, "r");
3.  Varcontenu ="";
4.  while (!feof(Varfichierouvert)) {
5.   Varcontenu .= fread(Varfichierouvert, 8192);
6.  }
7.  fclose (Varfichierouvert);
8.  if (VarOverwrite==true) {
9.  Varfichierouvert = fopen(Vardestination, "w+");
10. }else {
11. Varfichierouvert = fopen(Vardestination, "a+");
12. }
13. if ( !fwrite(Varfichierouvert, Varcontenu)) {
14.  echo "Impossible d'écrire dans le fichier". Varfilename;
15.  exit;
16. }
17. fclose (Varfichierouvert);
18. }
```

2.4.1 Donner la signification de la variable suivante :

Overwrite

2.4.2 Ecrivez le code faisant appel à la fonction MyFileCopy en écrivant 2 appels possibles pour un même nom de fichier et commenter ce code.

2.4.3 Expliquez la transmission des variables par référence ou par valeur.

2.5 EXERCICE N° 5

```
1. public static FctU2(int Vare){
2. if (Vare>0) {
3. Vare-1;
4. return 3*FctU2(Vare)+1;
5. } else  {
6. return 1;
7. }
8. }
```

2.5.1 Quel est le but de ce code ?

2.5.2 Corriger ce code.

2.5.3 Réécrivez cette fonction sans faire appel à la récursivité.

3 Problème (sur 8 points)

Il sera tenu compte de la clarté de la rédaction du code Java.
Vous prendrez toute initiative que vous jugerez nécessaire afin de pouvoir réaliser le sujet.
Toutefois, vous devrez les faire apparaître clairement dans votre devoir.

```
1. ...
2. fTabDisques=AjoutEnreg(fTabDisques, NoDisque, Prix, Editeur, Genre, Nom, Prenom, Titre,
   NbExemplaire);
3. echo ListeEnreg(fTabDisques);
4. echo NbEnreg(fTabDisques)."</br>";
5. echo NbAlbumJackson(fTabDisques)."</br>";
6. echo PrixMoyen(fTabDisques)."</br>";
7.  public static  AjoutEnreg(fTabDisques, NoDisque, Prix, Editeur, Genre, Nom, Prenom, Titre,
   NbExemplaire){
8. if (NoDisque!=""&&Prix>0){
9.  fLigneDisque["dis_id"]=NoDisque;
10. fLigneDisque["dis_prix"]=round(Prix, "2");
11. fLigneDisque["dis_edit"]=Editeur;
12. fLigneDisque["dis_genre"]=Genre;
13. fLigneDisque["dis_nom"]=strtoupper(Nom);
14. fLigneDisque["dis_prenom"]=Prenom;
15. fLigneDisque["dis_titre"]=Titre;
16. fLigneDisque["dis_nbex"]=NbExemplaire;
17. fTabDisques[]=fLigneDisque;
18. }
```

```
19. return fTabDisques;
20. }
21. public static ListeEnreg(fTabDisques) {
22. // Trier la table
23. foreach (fTabDisques as fLigneDisque) {
24. if (fLigneDisque["dis_nom"]=="JACKSON") {
25. echo fLigneDisque["dis_genre"]."\t";
26. echo fLigneDisque["dis_nom"]."\t";
27. echo fLigneDisque["dis_titre"]."</br>";
28. }
29. }
30. }
31. public static NbEnreg() {
32. NbEnreg=sizeof(fTabDisques);
33. return NbEnreg;
34. }
35. public static NbAlbumJackson(fTabDisques) {
36. foreach (fTabDisques as fLigneDisque) {
37. if (fLigneDisque["dis_nom"]="JACKSON") {
38. NbEnreg++;
39. }
40. }
41. return NbEnreg
42. }
43. public static PrixMoyen(fTabDisques) {
44. PrixTotal=0;
45. i=0;
46. foreach (fTabDisques as fLigneDisque) {
47. i++;
48. PrixTotal=PrixTotal+fLigneDisque["dis_prix"];
49. }
50. return PrixTotal/i;
51. }
```

3.1.1 Quel est l'objectif de cet extrait de programme ?

3.1.2 Corriger ce code afin qu'il s'exécute correctement

3.1.3 Critiquer ce code et proposer des améliorations (sans réécrire le code)

3.1.4 Ecrire les parties de programme qui permettent :

- De supprimer un enregistrement après avoir demandé confirmation
- De se déplacer vers l'enregistrement suivant lors de l'appui sur la commande « Suivant »
- De calculer la proportion de disques (en %) dont le nom est « JACKSON »,de stocker le résultat dans une variable et d'afficher le résultat
- De calculer le coût total des exemplaires de l'album dont le titre est « Thriller ».

Recommandations :

Vous devez choisir un langage et un seul, parmi ceux proposés, dans lequel vous composerez obligatoirement l'intégralité de votre devoir et vous indiquerez le langage choisi au début de votre copie. **Si cette règle n'est pas respectée, votre copie ne sera pas prise en compte.**

Toutes les réponses doivent figurer dans la copie (ne pas écrire directement sur le sujet). Pour les parties relatives aux corrections de code vous voudrez bien indiquer le numéro des lignes à ajouter (ligne précédente suivie de a, b, c ...), à modifier ou à supprimer de la manière suivante :

7a *instruction à jouter*
7b *instruction à ajouter*
Supprimer ligne 8
9 *instruction modifiée*

PHP

1 Questions (4 points)

1.1 Question n° 1

Citer tous les types d'itérations possibles associés à ce langage et justifier l'intérêt de choisir chaque type. Donner les syntaxes correspondantes.

1.2 Question n° 2

Qu'est-ce qu'une fonction récursive ?

Dans quels cas l'utiliser ?

Pourquoi l'utilisation de telles fonctions n'est elle pas toujours recommandée ? (expliciter votre réponse)

Qu'est-ce qu'une fonction récursive terminale ?

2 Exercices (8 points)

2.1 Exercice n° 1

```
1. <?php
2. if(isset($_POST["fVente1"]) && $_POST["fVente1"]!="")
3. {
4. $Vente1=$_POST["fVente1"];
5. $Vente2=$_POST["f Vente2"];
6. $ Vente3=$_POST["f Vente3"];
7. $Remise=$_POST["fRemise"];
```

```
8. }
9. $Total = GetTotal($ Vente1,$ Vente2, $ Vente3);
10. VentesTaxe($Total, $Remise);
11. function GetTotal($Vente1, $Vente2, $Vente3) {
12. $Total = $ Vente1 + $ Vente2 + $ Vente3;
13. return $Total ;
14. }
15. function VentesTaxe($Total, $Remise) {
16. $VentesTaxe = ($Total * .055}
17. $VentesTaxe = $VentesTaxe - {$Remise * $Total)
18. echo "Taxation totale :  ".$VentesTaxe
19. }
20. php?>
```

2.1.1 Quel est le but de ce code ?

2.1.2 Décrire la structure de ce code

2.1.3 Décrire sommairement les instructions présentes dans ce code

2.1.4 Corriger ce code afin que les procédures s'exécutent correctement

2.1.5 Critiquer ce code et proposer des améliorations (sans réécrire le code)

2.2 EXERCICE N° 2

```
1. <?php
2. if(isset($_POST["fPassword"]) && $_POST["fPassword"]!="")
3. {
4. $Login=$_POST[“fLogin”];
5. $Password=$_POST["fPassword"];
6. if ($Password=="password") {
7. $_SESSION[“LoginSucceeded”]=True ;
8. $_SESSION[“Login”]=$Login ;
9. $_SESSION[“Password”]=$Password ;
10. }
11. else {
12. echo "Mot de passe non valide, r&eacute;essayez !";
13. }
14. ?>
```

2.2.1 Quel est le but de ce code ?

2.2.2 A quoi sert l'instruction suivante :

$_SESSION[“LoginSucceeded ”]=True ;

2.2.3 Ajouter le code nécessaire à la bonne exécution de ce programme :

2.2.4 Critiquer ce code ainsi corrigé et proposer des améliorations (sans réécrire le code)

2.3 EXERCICE N° 3

```
1. Function IsMultiLine($Buffer, $UnixConvention) {
2. $LineSeparator=chr(10).chr(13);
3. If ($UnixConvention) {
4. $LineSeparator=chr(10);
5. }
6. if(ereg($LineSeparator,$Buffer)){
7. return true;
8. }
9. return false;
10. }
```

2.3.1 Quel est le but de ce code ?

2.3.2 A quoi sert la fonction suivante :

ereg (string $pattern , string $string [, array &$regs])

2.4 EXERCICE N° 4

L'extrait du code PHP présenté ci-dessous permet de copier un fichier source vers une destination

```
1. Function MyFileCopy($source, $destination, $Overwrite) {
2. $fichierouvert = fopen($source, "r");
3. $contenu ="";
4. while (!feof($fichierouvert)) {
5. $contenu .= fread($fichierouvert, 8192);
6. }
7. fclose ($fichierouvert);
8. if ($Overwrite==true) {
9. $fichierouvert = fopen($destination, "w+"};
10. }else {
11. $fichierouvert = fopen($destination, "a+");
12. }
13. if ( !fwrite($fichierouvert, $contenu)) {
14. echo "Impossible d'écrire dans le fichier". $filename;
15. exit;
16. }
17. fclose ($fichierouvert);
18. }
```

2.4.1 Donner la signification de la variable suivante :

$Overwrite

2.4.2 Ecrivez le code faisant appel à la fonction MyFileCopy en écrivant 2 appels possibles pour un même nom de fichier et commenter ce code.

2.4.3 Expliquez la transmission des variables par référence ou par valeur.

2.5 EXERCICE N° 5

```
1. function FctU2($e){
2. if ($e>0) {
3. $e-1;
4. return 3*FctU2($e)+1;
5. } else {
6. return 1;
7. }
8. }
```

2.5.1 Quel est le but de ce code ?

2.5.2 Corriger ce code.

2.5.3 Réécrivez cette fonction sans faire appel à la récursivité.

3 Problème (8 points)

Il sera tenu compte de la clarté de la rédaction du code 4GL
Vous prendrez toute initiative que vous jugerez nécessaire afin de pouvoir réaliser le sujet.
Toutefois, vous devrez les faire apparaître clairement dans votre devoir.

```
1. $fTabDisques=AjoutEnreg($fTabDisques, $NoDisque, $Prix, $Editeur, $Genre, $Nom,
   $Prenom, $Titre, $NbExemplaire);
2. echo ListeEnreg($fTabDisques);
3. echo NbEnreg($fTabDisques)."</br>";
4. echo NbAlbumJackson($fTabDisques)."</br>";
5. echo PrixMoyen($fTabDisques)."</br>";
6. function AjoutEnreg($fTabDisques, $NoDisque, $Prix, $Editeur, $Genre, $Nom, $Prenom,
   $Titre, $NbExemplaire){
7. if ($NoDisque!=""&&$Prix>0){
8. $fLigneDisque["dis_id"]=$NoDisque;
9. $fLigneDisque["dis_prix"]=round($Prix, "2");
10. $fLigneDisque["dis_edit"]=$Editeur;
11. $fLigneDisque["dis_genre"]=$Genre;
12. $fLigneDisque["dis_nom"]=strtoupper($Nom);
13. $fLigneDisque["dis_prenom"]=$Prenom;
14. $fLigneDisque["dis_titre"]=$Titre;
15. $fLigneDisque["dis_nbex"]=$NbExemplaire;
16. $fTabDisques[]=$fLigneDisque;
17. }
18. return $fTabDisques;
19. }
20. function ListeEnreg($fTabDisques) {
21. foreach ($fTabDisques as $fLigneDisque) {
22. if ($fLigneDisque["dis_nom"]=="JACKSON") {
23. echo $fLigneDisque["dis_genre"]."\t";
24. echo $fLigneDisque["dis_nom"]."\t";
25. echo $fLigneDisque["dis_titre"]."</br>";
26. }
27. }
28. }
29. function NbEnreg() {
30. $NbEnreg=sizeof($fTabDisques);
31. return $NbEnreg;
32. }
```

```
33. function NbAlbumJackson($fTabDisques) {
34. foreach ($fTabDisques as $fLigneDisque) {
35. if ($fLigneDisque["dis_nom"]="JACKSON") {
36. $NbEnreg++;
37. }
38. }
39. return $NbEnreg;
40. }
41. function PrixMoyen($fTabDisques) {
42. $PrixTotal=0;
43. $i=0;
44. foreach ($fTabDisques as $fLigneDisque) {
45. $i++;
46. $PrixTotal=$PrixTotal+$fLigneDisque["dis_prix"];
47. }
48. return $PrixTotal/$i;
49. }
```

3.1.1 Quel est l'objectif de cet extrait de programme ?

3.1.2 Corriger ces fonctions afin qu'elles s'exécutent correctement

3.1.3 Critiquer ce code et proposer des améliorations (sans réécrire le code)

3.1.4 Ecrire les parties de programme qui permettent :

- De supprimer un enregistrement
- De se déplacer vers l'enregistrement suivant, lors de l'appui sur la commande « Suivant »
- De calculer la proportion de disques (en %) dont le nom est « JACKSON », de stocker le résultat dans une variable et d'afficher le résultat
- De calculer le coût des exemplaires de l'album dont le titre est « Thriller ».

Recommandations :

Vous devez choisir un langage et un seul, parmi ceux proposés, dans lequel vous composerez obligatoirement l'intégralité de votre devoir et vous indiquerez le langage choisi au début de votre copie. **Si cette règle n'est pas respectée, votre copie ne sera pas prise en compte.**

Toutes les réponses doivent figurer dans la copie (ne pas écrire directement sur le sujet). Pour les parties relatives aux corrections de code vous voudrez bien indiquer le numéro des lignes à ajouter (ligne précédente suivie de a, b, c ...), à modifier ou à supprimer de la manière suivante :

7a *instruction à jouter*
7b *instruction à ajouter*
Supprimer ligne 8
9 *instruction modifiée*

XSLT

1 Questions (4 points)

1.1 Question n° 1

Citez toutes les structures de contrôle de XSLT. Expliquer les possibilités offertes par ce langage pour effectuer des itérations ou boucles.

1.2 Question n° 2

Expliquer les notions de variables et de paramètres en XSLT. Quelles sont les différences avec les langages de programmation procéduraux ?

Expliquer la manière de construire un modèle (ou template) récursif en XSLT ?

Qu'est-ce qu'une fonction récursive terminale ? Que peut-on dire des templates récursifs de XSLT ?

2 Exercices

2.1 Exercice n° 1

Soit la DTD suivante (ventes.dtd) :

```
1. <?xml version="1.0" encoding="iso-8859-1" ?>
2. <!ELEMENT ventes    (vente*)>
3. <!ELEMENT vente  (numero, prix)>
4. <!ATTLIST vente paiement (cheque | carte | traite) "cheque">
5. <!ELEMENT numero      (#PCDATA)>
6. <!ELEMENT prix      (#PCDATA)>
```

Soit le document XML suivant :

```
1. <?xml version="1.0" encoding="ISO-8859-1"?>
2. <!DOCTYPE ventes SYSTEM "ventes.dtd">
3. <ventes>
4. <vente paiement="carte">
5. <numero>1</numero>
6. <prix>34</prix>
7. </vente>
8. <vente>
9. <prix>56</prix>
10. </vente>
11. <vente paiement="carte">
12. <numero>3</numero>
13. <prix>23</prix>
14. </vente>
15. </ventes>
```

2.1.1 Expliquer le code de la ligne 2 du document XML

2.1.2 Est-ce que ce document est bien formé?

2.1.3 Est-ce que ce document est valide?

2.1.4 Donnez les expressions XPath correspondant aux requêtes suivantes

a. les numéros de toutes les ventes
b. les numéros des ventes dont le prix est supérieur à 30
c. les numéros des ventes dont le paiement est par carte
d. le pourcentage de ventes dont le paiement est en carte

2.2 EXERCICE N° 2

Etant donnée la DTD de l'exercice n°1, considérer le code suivant

```
1. <?xml version="1.0"?>
2. <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
3. <xsl:output method="html" encoding="Windows-1252" />
4. <xsl:template match="/ventes">
5. <xsl:variable name="total">
6. <xsl:call-template name="Calcul">
7. <xsl: param name="listePrix"
8. select="/ventes/vente/prix"/>
9. </xsl:call-template>
10. </xsl:variable>
11. <xsl:text>Total des ventes: </xsl:text>
12. <xsl:value-of select="$total"/>
13. </xsl:template>
14. <xsl:template name="Calcul">
```

```
15. <xsl:param name="listePrix">
16. <xsl:choose>
17. <xsl:when test="$listePrix">
18. <xsl:variable name="v" select="$listePrix[1]"/>
19. <xsl:variable name="autresPrix">
20. <xsl:call-template name="Calcul">
21. <xsl:with-param name="listePrix"
22. select="$listePrix[position() != 1]"/>
23. </xsl:call-template>
24. </xsl:variable>
25. <xsl:value-of select="$v + $autresPrix"/>
26. </xsl:when>
27. <xsl:otherwise>0</xsl:otherwise>
28. </xsl:choose>
29. </xsl:template>
30. </xsl:stylesheet>
```

2.2.1 Quel est le but de ce code ?

2.2.2 Décrire la structure de ce code

2.2.3 Décrire sommairement les instructions présentes dans ce code

2.2.4 Corriger ce code afin que les procédures s'exécutent correctement

2.3 EXERCICE N° 3

Soit le fichier XML suivant

```
1.  <?xml version="1.0" ?>
2.  <?xml-stylesheet href="persons.xsl" type="text/xsl" ?>
3.  <!DOCTYPE personnes SYSTEM "personnes.dtd">
4.  <personnes>
5.   <personne username="MD" password="a1b2cD">
6.  <nom>Dupond</nom>
7.  <prenom>Michel</prenom>
8.  <prenom>Pierre</prenom>
9.  <email>m.dupond@gmail.com</email>
10. </personne>
11. <personne username="MI1" password="6xyTr">
12. <nom>Rauffas</nom>
13. <prenom>Marie</prenom>
14. <prenom>Isabelle</prenom>
15. <email>mar@hotmail.com</email>
16. </personne>
17. <personne username="DD" password="oP4erA">
18. <nom>Durand</nom>
```

```
19. <prenom>Didier</prenom>
20. <email>dede@orange.fr</email>
21. </personne>
22. </personnes>
```

Considérons la feuille XSLT suivante:

```
1.  <?xml version="1.0" encoding="UTF-8"?>
2.  <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
3.  <xsl:output method="xml" indent="yes"/>
4.  <xsl:template match="/personnes">
5.  <users>
6.  <xsl:apply-templates select="personne"/>
7.  </users>
8.  </xsl:template>
9.  <xsl:template match="personne">
10. <user>
11. <username>
12. <xsl:value-of select="@username" />
13. </username>
14. <password>
15. <xsl:value-of select="@password" />
16. </password>
17. </user>
18. </xsl:template>
19. </xsl:stylesheet>
```

2.3.1 Quel est le but de ce code ?

2.3.2 Ecrire la DTD du fichier XML original

2.3.3 Déduire la DTD du document XML produit par ce XSLT

2.4 EXERCICE N° 4

```
1.  <xsl:template name="remplacer">
2.  <xsl:param name="originale"/>
3.  <xsl:param name="recherche"/>
4.  <xsl:param name="remplacement" select="''"/>
5.  <xsl:variable name="premier">
6.  <xsl:choose>
7.  <xsl:when test="contains($originale, $recherche)">
8.  <xsl:value-of select="substring-before($originale, $recherche)"/>
9.  </xsl:when>
10. <xsl:otherwise>
11. <xsl:value-of select="$originale"/>
```

```
12. </xsl:otherwise>
13. </xsl:choose>
14. </xsl:variable>
15. <xsl:variable name="milieu">
16. <xsl:choose>
17. <xsl:when test="contains($originale, $recherche)">
18. <xsl:value-of select="$remplacement"/>
19. </xsl:when>
20. <xsl:otherwise>
21. <xsl:text></xsl:text>
22. </xsl:otherwise>
23. </xsl:choose>
24. </xsl:variable>
25. <xsl:variable name="dernier">
26. <xsl:choose>
27. <xsl:when test="contains($originale, $recherche)">
28. <xsl:choose>
29. <xsl:when test="contains(substring-after($originale, $recherche), $recherche)">
30. <xsl:call-template name="remplacer">
31. <xsl:with-param name="originale">
32. <xsl:value-of select="substring-after($originale, $recherche)"/>
33. </xsl:with-param>
34. <xsl:with-param name="recherche">
35. <xsl:value-of select="$recherche"/>
36. </xsl:with-param>
37. <xsl:with-param name="remplacement">
38. <xsl:value-of select="$remplacement"/>
39. </xsl:with-param>
40. </xsl:call-template>
41. </xsl:when>
42. <xsl:otherwise>
43. <xsl:value-of select="substring-after($originale, $recherche)"/>
44. </xsl:otherwise>
45. </xsl:choose>
46. </xsl:when>
47. <xsl:otherwise>
48. <xsl:text></xsl:text>
49. </xsl:otherwise>
50. </xsl:choose>
51. </xsl:variable>
52. <xsl:value-of select="concat($premier, $milieu, $dernier)"/>
53. </xsl:template>
```

2.4.1 Quel est le but de ce code ? Expliquer le code.

2.4.2 A quoi sert la fonction suivante :

```
<xsl:when test="contains($originale, $recherche)">
<xsl:value-of select="concat($premier, $milieu, $dernier)"/>
```

2.4.3 Compléter ce code pour en faire une feuille XSLT qui utilise le template donné. Proposer un exemple d'utilisation.

2.5 EXERCICE N° 5

Soit le fichier XML :

```
1.  <?xml version="1.0"?>
2.  <mesure>
3.  <mesure>
4.  <mesure_name></mesure_name>
5.  <mesure_name>Mesure1</mesure_name>
6.  <mesure_desc></mesure_desc>
7.  <mesure_desc>Mesure Desc1</mesure_desc>
8.  <mesure_desc>Mesure Desc2</mesure_desc>
9.  </mesure>
10. <mesure>
11. <mesure_name></mesure_name>
12. <mesure_desc></mesure_desc>
13. </mesure>
14. </mesures>
```

Considérons le code XSLT :

```
1.  <?xml version="1.0" encoding="UTF-8"?>
2.  <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
3.  <xsl:output method="xml" indent="yes"/>
4.  <xsl:template match="/mesures">
5.  <xsl:copy>
6.  <xsl:apply-templates select="mesure"/>
7.  </xsl:copy>
8.  </xsl:template>
9.  <xsl:template match="mesure">
10. <xsl:copy>
11. <xsl:apply-templates select="*"/>
12. </xsl:copy>
13. </xsl:template>
14. <xsl:template match="*">
15. <xsl:if test="not(preceding-sibling::*[text() != '' and
16. name() = name(current())]) and
17. text() != ''">
18. <xsl:copy-of select="."/>
```

```
19. </xsl:if>
20. </xsl:template>
21. </xsl:stylesheet>
```

2.5.1 Quel est le but de ce code ?

2.5.2 Que produit ce code pour la deuxième mesure ?

2.5.3 Réécrivez ce code pour obtenir un résultat pour la seconde mesure.

3 Problème

Soit la DTD suivante décrivant la structure d'une commande :

```
1. $<?xml version="1.0" encoding="iso-8859-1" ?>
2. <!ELEMENT commandes    (commande*)>
3. <!ELEMENT commande  (client,produits)>
4. <!ATTLIST commande id CDATA #REQUIRED date CDATA #REQUIRED>
5. <!ELEMENT client   (adresse+)>
6. <!ATTLIST client id CDATA #REQUIRED >
7. <!ELEMENT adresse    (contact*, rue*, ville*, code-postal*)>
8. <!ATTLIST adresse type (entreprise | livraison) "entreprise">
9. <!ELEMENT contact    (titre, nom, prenom)>
10. <!ELEMENT titre    (#PCDATA)>
11. <!ELEMENT nom    (#PCDATA)>
12. <!ELEMENT prenom    (#PCDATA)>
13. <!ELEMENT rue    (#PCDATA)>
14. <!ELEMENT ville (#PCDATA)>
15. <!ELEMENT code-postal (#PCDATA)>
16. <!ELEMENT produits   (produit*)>
17. <!ELEMENT produit    (nom, quantite, prix)>
18. <!ATTLIST produit reference CDATA #REQUIRED>
19. <!ELEMENT quantite    (#PCDATA)>
20. <!ELEMENT prix    (#PCDATA)>
```

Deux exemples de fichiers commandes (commande1.xml et commande2.xml) sont donnés en annexe. Nous nous proposons de traiter l'ensemble des commandes en créant le document maître suivant :

```
1. <?xml version="1.0"?>
2. <commandes>
3. <bon filename="commande1.xml"/>
4. <bon filename="commande2.xml"/>
5. </commandes>
```

Soit le code XSLT suivant :

```
1.  <?xml version="1.0"?>
2.  <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
3.  <xsl:output method="html" indent="no"/>
4.  <xsl:strip-space elements="*"/>
5.  <xsl:template match="/">
6.  <html>
7.  <head>
8.  <title><xsl:value-of select="/commandes/titre"/></title>
9.  </head>
10. <body>
11.  <xsl:for-each select="/commandes/bon">
12. <xsl:apply-templates select="document(@filename)/commande"/>
13. </xsl:for-each>
14. </body>
15. </html>
16. </xsl:template>
17. <xsl:template match="commande">
18. <h1>
19. <xsl:value-of select="client/adresse[@type='entreprise']/contact/titre"/>
20. <xsl:text> </xsl:text>
21. <xsl:value-of select="client/adresse[@type='entreprise']/contact/nom"/>
22. <xsl:text> </xsl:text>
23. <xsl:value-of select="client/adresse[@type='entreprise']/contact/prenom"/>
24. </h1>
25. <p>
26. <xsl:text>Commande du </xsl:text>
27. <xsl:value-of select="@date"/>
28. </p>
29. <h2>Produits:</h2>
30. <table width="100%" border="1" cols="15% 55% 10% 10% 10%">
31. <tr bgcolor="lightgreen">
32. <th>Référence</th>
33. <th>Produit</th>
34. <th>Quantité</th>
35. <th>Prix unitaire</th>
36. <th>Montant</th>
37. </tr>
38. <xsl:for-each select="produits/produit">
39. <tr>
40. <xsl:attribute name="bgcolor">
41. <xsl:choose>
42. <xsl:when test="position() mod 2">
43. <xsl:text>white</xsl:text>
44. </xsl:when>
45. <xsl:otherwise>
```

```
46. <xsl:text>cyan</xsl:text>
47. </xsl:otherwise>
48. </xsl:choose>
49. </xsl:attribute>
50. <td>
51. <xsl:value-of select="@reference"/>
52. </td>
53. <td>
54. <b><xsl:value-of select="nom"/></b>
55. </td>
56. <td align="center">
57. <xsl:value-of select="quantite"/>
58. </td>
59. <td align="right">
60. <xsl:value-of select="prix"/>
61. </td>
62. <td align="right">
63.  <xsl:choose>
64. <xsl:when test="position()=1">
65. <xsl:value-of select="format-number(prix * quantite, '# ###,00 Euros')"/>
66. </xsl:when>
67. <xsl:otherwise>
68. <xsl:value-of select="format-number(prix * quantite, '# ###,00 Euros')"/>
69. </xsl:otherwise>
70. </xsl:choose>
71. </td>
72. </tr>
73. </xsl:for-each>
74. <tr>
75. </table>
76. </xsl:template>
77. </xsl:stylesheet>
```

3.1.1 Quel est l'objectif de ce code ?

3.1.2 Expliquer la structure de ce code

3.1.3 Compléter ce code pour calculer le total de la commande de chaque client

---

Commande1.xml

```
1. <?xml version="1.0"?>
2. <!DOCTYPE commande SYSTEM "commande.dtd">
3. <commande id="0001" date="12-08-2008">
4. <client id="4738">
```

```
5.  <adresse type="entreprise">
6.  <contact>
7.  <titre>M.</titre>
8.  <nom>Dupuis</nom>
9.  <prenom>Jacques</prenom>
10. </contact>
11. <rue>12 Rue Rivoli</rue>
12. <ville>Paris</ville>
13. <code-postal>75000</code-postal>
14. </adresse>
15. <adresse type="livraison"/>
16. </client>
17. <produits>
18. <produit reference="28392-33-CN">
19. <nom>Chocolat noir</nom>
20. <quantite>1</quantite>
21. <prix>35.60</prix>
22. </produit>
23. <produit reference="28813-70-SV">
24. <nom>Salade verte</nom>
25. <quantite>3</quantite>
26. <prix>2.50</prix>
27. </produit>
28. </produits>
29. </commande>
```

Commande2.xml

```
1.  <?xml version="1.0"?>
2.  <!DOCTYPE commande SYSTEM "commande.dtd">
3.  <commande id="9002" date="07-12-2008">
4.  <client id="3345">
5.  <adresse type="entreprise">
6.  <contact>
7.  <titre>M.</titre>
8.  <nom>Dufour</nom>
9.  <prenom>Michel</prenom>
10. </contact>
11. <rue>5 rue C De Gaulle</rue>
12. <ville>Toulouse</ville>
13. <code-postal>31000</code-postal>
14. </adresse>
15. <adresse type="livraison"/>
16. </client>
17. <produits>
18. <produit reference="68319-99-CF">
```

```
19. <nom>Couscous fin</nom>
20. <quantite>2</quantite>
21. <prix>5.20</prix>
22. </produit>
23. <produit reference="89652-98-VA">
24. <nom>Viande agneau</nom>
25. <quantite>3</quantite>
26. <prix>15.85</prix>
27. </produit>
28. <produit reference="34218-22-TR">
29. <nom>Tulipe rouge</nom>
30. <quantite>10</quantite>
31. <prix>2.30</prix>
32. </produit>
33. </produits>
34. </commande>
```

Recommandations :

Vous devez choisir un langage et un seul, parmi ceux proposés, dans lequel vous composerez obligatoirement l'intégralité de votre devoir et vous indiquerez le langage choisi au début de votre copie. **Si cette règle n'est pas respectée, votre copie ne sera pas prise en compte.**

Toutes les réponses doivent figurer dans la copie (ne pas écrire directement sur le sujet). Pour les parties relatives aux corrections de code vous voudrez bien indiquer le numéro des lignes à ajouter (ligne précédente suivie de a, b, c ...), à modifier ou à supprimer de la manière suivante :

7a *instruction à jouter*
7b *instruction à ajouter*
Supprimer ligne 8
9 *instruction modifiée*

VISUAL BASIC 6

1 Questions (4 points)

1.1 QUESTION N° 1

Citez tous les types d'itération possibles associés à ce langage et justifier l'intérêt de choisir chaque type. Donner les syntaxes correspondantes.

1.2 QUESTION N° 2

Qu'est-ce qu'une fonction récursive ?

Dans quels cas l'utiliser ?

Pourquoi il n'est pas toujours conseillé d'utiliser de telles fonctions ? (expliciter votre réponse)

Qu'est-ce qu'une fonction récursive terminale ?

2 Exercices (8 points)

2.1 EXERCICE N° 1

```
1. Private Sub GetTotal()
2. Dim curTotal As Integer
3. curTotal = txtSale1.Text + txtSale2.Text + txtSale3.txt
4. Call SalesTax(curTotal, sngRemise)
5. End Sub
6. Public Sub SalesTax(curTotal As Currency, sngRateRemise As Single)
7. Dim curSalesTax As Currency
8. Dim intMsg As Integer  ' For MsgBox()
9. curSalesTax = (curTotal * .055)
```

```
10. curSalesTax = curSalesTax - (sngRateRemise * curTotal)
11. intMsg = MsgBox("Taxation totale :  & curSalesTax)
12. End Sub
```

2.1.1 Quel est le but de ce code ?

2.1.2 Décrire la structure de ce code

2.1.3 Décrire sommairement les instructions présentes dans ce code

2.1.4 Corriger ce code afin que les procédures s'exécutent correctement

2.1.5 Critiquer ce code et proposer des améliorations (sans réécrire le code).

2.2 EXERCICE N° 2

```
1. Option Explicit
2. Private Sub cmdCancel_Click()
3. LoginSucceeded = False
4. Me.Hide
5. End Sub
6. Private Sub cmdOK_Click()
7. If txtPassword = "password"
8. Then LoginSucceeded = True ; Me.Hide
9. Else
10. MsgBox "Mot de passe non valide, réessayez !", , "Connexion"
11. txtPassword.SetFocus
12. SendKeys "{Home}+{End}"
13. End Sub
```

2.2.1 Quel est le but de ce code ?

2.2.2 Donner la signification de l'instruction suivante :

Option Explicit

2.2.3 Ajouter le code nécessaire à la bonne exécution de ce code.

2.2.4 Critiquer ce code ainsi corrigé et proposer des améliorations (sans réécrire le code).

2.3 EXERCICE N° 3

```
1. Public Function IsMultiLine(ByVal Buffer As String, Optional ByVal UnixConvention As Boolean)
   As Boolean
2. Dim LineSeparator As String
3. LineSeparator = vbCrLf
4. If UnixConvention Then
5. LineSeparator = vbLf
6. End If
7. IsMultiLine = (InStr(Buffer, LineSeparator) > 0)
8. End Function
```

2.3.1 Quel est le but de ce code ?

2.3.2 Donner la signification du terme suivant :

Optional

2.4 EXERCICE N° 4

L'extrait du code Visual Basic présenté ci-dessous permet de copier un fichier source vers une destination

```
1. Option Explicit
2. Private Declare Function CopyFileA Lib "kernel32" (ByVal lpExistingFileName As String, ByVal
   lpNewFileName As String, ByVal bFailIfExists As Long) As Long
3. Private Const MAX_PATH = 260
4. Function MyFileCopy(ByVal Source As String, ByVal Destination As String, Optional ByVal
   Overwrite As Boolean = True) As Boolean
5. If Len(Destination) > MAX_PATH Or Len(Source) > MAX_PATH Then
6. MyFileCopy = False
7. Else
8. If (Overwrite) Then
9. MyFileCopy = (CopyFileA(Source, Destination, 0) <> 0)
10. Else
11. MyFileCopy = (CopyFileA(Source, Destination, 1) <> 0)
12. End If
13. End If
14. End Function
```

2.4.1 Donner la signification de la variable suivante :

Overwrite

2.4.2 Créer la procédure faisant appel à la fonction MyFileCopy en écrivant 3 appels possibles pour un même nom de fichier et commenter le code.

2.4.3 Expliquez la transmission des variables par référence ou par valeur.

2.5 EXERCICE N° 5

```
1. Private Function FctU2 (e As Integer) As Long
2. Dim temp As Long
3.  e = e-1
4.  temp = 3 * FctU2(e) +1
5.  FctU2 = temp
6. End Function
```

2.5.1 Quel est le but de ce code ?

2.5.2 Corriger ce code.

2.5.3 Réécrivez cette fonction sans faire appel à la récursivité.

3 Problème (8 points)

Il sera tenu compte de la clorté de la rédaction du code Visual Basic 6.
Vous prendrez toute initiative que vous jugerez nécessaire afin de pouvoir réaliser le sujet.
Toutefois, vous devrez les foire apparoître clairement dans votre devoir.
Vous devrez privilégier l'utilisation des objets ADO .

Le code ci-dessous est un extrait d'une application.

```
1. Option Explicit
2. Dim ma_BD As ADODB.Connection
3. Dim mon_enregistrement As ADODB.Recordset
4. Sub Afficher_JACKSON_Click()
5. mon_enregistrement.Open "SELECT * FROM Disque WHERE nom = 'JACKSON' ORDER BY_
   'Genre_musical, nom'", ma_BD, adOpenKeyset, adLockOptimistic
6. Call liste_enregistrement_jackson()
7. Close
8. End Sub
```

```
1. Sub liste_enregistrement_jackson(enreg As Recordset)
2. Dim liste As String
3. enreg.MoveFirst
4. Do
5. liste = liste & enreg!Numero_disque & ", " & enreg!Prix_unitaire & ", " & _
   enreg!Numero_Maison_Disque & ", " & _
6. enreg!Genre_musical & ", " & enreg!Nom & ", " & enreg!Prenom & ", " & enreg!Titre_album
   & ", " _ & enreg!Nombre_exemplaire & vbCrLf
7. enreg.MoveNext
8. Loop While enreg.EOF
9. MsgBox liste, , "Liste des albums de ""JACKSON"""
10. enreg.Filter = ""
11. End Sub
```

```
1. Sub Nombre_exemplaire_Disque_Click()
2. mon_enregistrement.Open "SELECT * FROM Disque", ma_BD, adOpenKeyset,
   adLockOptimistic
3. txtCompteur = nb_disque(mon_enregistrement)
4. Close
5. End Sub
```

```
1. Function nb_disque(enreg As Recordset) As Integer
2. enreg.MoveFirst
3. enreg.MoveLast
4. nb_disque =RecordCount
5. End Function
```

```
1. Sub Nbre_Disques_Jackson_Click()
2. mon_enregistrement.Open "SELECT * FROM Disque WHERE nom = 'JACKSON'", ma_BD,
   adOpenKeyset, adLockOptimistic
3. txtCompteurJackson = nb_disque_Jackson()
4. Close
5. End Sub
```

```
1. Function nb_disque_Jackson(enreg As Recordset) As Integer
2. enreg.MoveFirst
3. enreg.MoveLast
4. nb_disque_Jackson =RecordCount
5. End Function
```

```
1. Sub Prix_unitaire_Moyen_click()
2. Dim a As String
3. Set mon_enregistrement = New ADODB.Recordset
4. mon_enregistrement.Open "SELECT AVG (Disque.Prix_unitaire) AS Prixmoyen FROM
   DISQUE", ma_BD, adOpenKeyset, adLockOptimistic
5. txtPrixMoyen = mon_enregistrement!Prix_moyen
6. End Sub
```

```
1. Sub Form_Load()
2. Set ma_BD = New ADODB.Connection
3. ma_BD.Provider = "Microsoft.Jet.OLEDB.4.0"
4. ma_BD.ConnectionString = "C:\Users\Toto\Documents\DISQUE.mdb"
5. ma_BD.Open
6. Set mon_enregistrement = New ADODB.Recordset
7. End Sub
```

```
1. Sub Form_Load()
2. Set ma_BD = New ADODB.Connection
3. ma_BD.Provider = "Microsoft.Jet.OLEDB.4.0"
4. ma_BD.ConnectionString = "C:\Users\Toto\Documents\DISQUE.mdb"
5. ma_BD.Open
6. Set mon_enregistrement = New ADODB.Recordset
7. ma_BD.mon_enregistrement.Open ("Disque")
8. Call Initialiser
9. End Sub
```

```
1.  Private Sub Initialiser()
2.  With Me
3.  .txtNumDisque = 0
4.  .txtPrix = 0
5.  .txtNumMaisonDisque = 0
6.  .txtGenreMusical = ""
7.  .txtNom = ""
8.  .txtPrenom = ""
9.  .txtTitreAlbum = ""
10. .txtNombreDisque = 0
11. End With
12. End Sub
```

```
1.  Sub cmdAjouter_click()
2.  Dim Requete As String
3.  With Me
4.  If Not IsNull(txtNumDisque) Then
5.  mon_enregistrement.Open "SELECT Count(Disque.nom) AS Nb_enreg * FROM Disque
    WHERE Numero_disque='" & .txtNumDisque & "'  AND Genre_musical= '" &
    .txtGenreMusical & " ' AND Nom='" & .txtNom & " ' AND Prenom='" & .txtPrenom & " ' AND
    Titre_album='" & .txtTitreAlbum & "'", ma_BD, adOpenKeyset, adLockOptimistic
6.  If mon_enregistrement!Nb_enreg = 0 Then
7.  mon_enregistrement.AddNew
8.  mon_enregistrement!Numero_disque = .txtNumDisque
9.  If .txtPrix < 0 Then
10. .txtPrix = 0
11. End If
12. mon_enregistrement!Prix_unitaire = Round(.txtPrix, 2)
13. mon_enregistrement!Numero_Maison_Disque = .txtNumDisque
14. mon_enregistrement!Genre_musical = .txtGenreMusical
15. mon_enregistrement!Nom = UCase(.txtNom)
16. mon_enregistrement!Prenom = .txtPrenom
17. mon_enregistrement!Titre_album = .txtTitreAlbum
18. mon_enregistrement!Nombre_exemplaire = 1
19. Update
20. Else
21. mon_enregistrement!Nombre_exemplaire =Nombre_exemplaire + .txtNombreDisque
22. If mon_enregistrement!Prix_unitaire < .txtPrix Then
23. mon_enregistrement!Prix_unitaire = .txtPrix
24. End If
25. Update
26. End If
27. End If
28. Close
```

29. End With
30. End Sub

3.1.1 Quel est l'objectif de ces procédures ?

3.1.2 Corriger ce code afin que les procédures s'exécutent correctement

3.1.3 Critiquer ce code et proposer des améliorations (sans réécrire le code).

3.1.4 Ecrire les parties de programme qui permettent :

- De supprimer un enregistrement après avoir demandé confirmation
- De se déplacer vers l'enregistrement suivant, lors de l'appui sur la commande « Suivant »
- De calculer la proportion de disques (en %) dont le nom est « JACKSON » stocker le résultat dans une variable et afficher le résultat
- De calculer le coût total des exemplaires de l'album dont le titre est « Thriller ».