NF01 – Printemps 2016

Examen Final

ATTENTION !
Utilisez quatre copies séparées, une par partie

Document autorisé : une feuille de notes A4 recto-verso **manuscrite.**

1ère Partie : tableaux et enregistrements *(5 points)*

On souhaite réaliser un système de proposition de mots à partir d'un dictionnaire et d'un jeu de lettres. Les lettres dont le joueur dispose seront traitées comme une chaine de caractères. Les valeurs des lettres seront accessibles dans un tableau d'entiers de type :

```
TValeurs = array['a'..'z'] of integer; //type pour les valeurs de lettres
```

On considérera que le tableau de valeurs des lettres est déjà saisi. Un mot du dictionnaire sera de type `MotDico`, correspondant à un enregistrement contenant le mot (de type `string`) et le nombre de points pour ce mot (de type `integer`). Le nombre de points sera calculé à partir des valeurs des lettres. Un dictionnaire sera de type `TDico`, un tableau de `NMAX` mots.

1) Ecrire les types `MotDico` et `TDico`.

2) Ecrire en Pascal une procédure de saisie des mots du dictionnaire.

   N.B. : La procédure demande à l'utilisateur de saisir des mots. La saisie s'arrête lorsque l'utilisateur saisit une chaine de caractères vide. La chaîne de caractères vide sera insérée à la fin du tableau pour permettre par la suite de détecter la fin du dictionnaire.

   En parallèle de la saisie, la procédure calculera le nombre de points associés au mot à partir du tableau de valeurs des lettres et mettra à jour cette valeur dans le dictionnaire.

   L'appel de la procédure se fera ainsi :

   ```
   SaisieDictionnaire(dictionnaire,valeurs);
   ```

3) Ecrire en Pascal une fonction qui calcule le nombre de points à partir d'un mot passé en paramètre (`mot`) venant du dictionnaire ainsi que d'un ensemble de lettres correspondant aux lettres dont le joueur dispose (`lettresJoueur`) et calcule le nombre de points pour ce mot. La fonction renverra 0 si le mot n'est pas possible avec les lettres du joueur.

   L'appel de la fonction se fera ainsi :

   ```
   points := PointsMot(mot,lettresJoueur);
   ```

4) Ecrire en Pascal une fonction qui renvoie le mot du dictionnaire qui vaut le plus de points.

```
const
   NMAX = 50;
```

```
type
    //tSTRING = array [1..NMAX] of string;
    Dico = record
          mot : string;
          points : integer;
          end;
    tDICO = array [1..NMAX] of Dico;
    tL = array ['a'..'z'] of integer;
var
   c : integer;
   tLettres : tL;
   tDictionnaire : tDico;
   meilleurMot, Lettres : string;


procedure ValeursLettres (var t : tL);
var
   l : char;
begin
     for l := 'a' to 'z' do
         t[l] := 1;
     for l := 'a' to 'z' do
         case l of
              'd', 'm', 'g' : t[l] := 2;
              'b', 'c', 'p' : t[l] := 3;
              'f', 'h', 'v' : t[l] := 4;
              'j', 'q'      : t[l] := 8;
              'k', 'w', 'x', 'y', 'z' : t[l] := 10;
         end;

//     for l := 'a' to 'z' do
//         writeln(l, ' : ', t[l]);

end;

function CalculValeurMot(mot : string ; tLettres : tL) : integer;
// Mot : copie du mot venant du dictionnaire
// tLettres : copie du tableau de valeur des lettres
var
   j, points : integer; // j : compteur pour avancer dans les lettres de
Mot, p : position de la lettre ŧ enlever dans les lettres du joueur
begin
      j := 1;
      points := 0;
      while (j<=length(mot)) do
            begin
                points := points + tLettres[mot[j]];
                j := j + 1;
             end;
     CalculValeurMot := points;
end;

procedure Saisie (tLettres : tL ; var tD : tDICO);
var
   i : integer;
```

```
begin
     writeln('----Saisie----');
     writeln('Saisissez les mots, rien pour arrÍter');
     i := 0;
     repeat
         i := i + 1;
         readln(tD[i].mot);
         tD[i].points := CalculValeurMot(tD[i].mot, tLettres);
     until tD[i].mot ='';
end;


procedure Affichage(tD : tDICO);
var
   i : integer;
begin
     writeln('----Affichage----');
     i := 1;
     while tD[i].mot <> '' do
           begin
                writeln(tD[i].mot, '(', tD[i].points, ' points)');
                i := i + 1;
           end;
end;


function MotPossible(Lettres : string ; mot : string) : boolean;
// Lettres : copie des Lettres du joueur
// mot : copie du mot du Dictionnaire
var
   j,p : integer; // j : compteur pour avancer dans les lettres de Mot, p :
position de la lettre ‡ enlever dans les lettres du joueur
   b : boolean; // booleen pour savoir si les lettres du mot sont dans les
lettres du joueur
begin
      j := 1;
      b := true;
      while (j<=length(mot)) and b do
      // pour chaque lettre du mot du dictionnaire et tant que le mot est
possible
            begin
               p := pos(mot[j], Lettres); // on cherche si la lettre du mot
est dans les lettres du joueur
               if p <> 0 then
                        delete(Lettres, p, 1) // enleve la lettre trouvee
de la copie des lettres du joueur (evite problemes si memes lettres
plusieurs fois
                else
                         b := false;
                j := j + 1;
             end;
     MotPossible := b;
end;
```

```
function CalculMeilleurMot (Lettres : string ; tDictionnaire : tDICO) :
string;
// Lettres : copie des lettres du joueur
// tDictionnaire : copie du dictionnaire
var
   i, iMax : integer;
begin
     i := 1;
     iMax := 1;
     while tDictionnaire[i].mot <> '' do
           begin
                writeln(tDictionnaire[i].mot, ' (',
                       tDictionnaire[i].points, ') - ', Lettres);
                if MotPossible(Lettres, tDictionnaire[i].mot) then
                   if tDictionnaire[i].points >= tDictionnaire[iMax].points
                     then
                      begin
                         iMax := i;
                         writeln('        Points max : ',
                          tDictionnaire[iMax].points,' , iMax : ',iMax);
                       end;
                i := i + 1;
           end;

          writeln('Le meilleur mot est a faire est : ',
           tDictionnaire[iMax].mot, ' - avec ', tDictionnaire[iMax].points,
            ' points');
     CalculMeilleurMot := tDictionnaire[iMax].mot;
end;

function Menu : integer;
begin
      writeln('-----------------MENU-------------------');
      writeln('0. Sortir');
      writeln('1. Saisie');
      writeln('2. Ajout Mots');
      writeln('3. Affichage');
      writeln('4. Calcul meilleur mot');
      readln(Menu);
end;

begin
     ValeursLettres(tLettres);
     repeat
          c := Menu;
          case c of
               1 : Saisie(tLettres, tDictionnaire);
               3 : Affichage(tDictionnaire);
               4 : begin
                    writeln('Saisie des lettres');
                    readln(Lettres);
                    meilleurMot := CalculMeilleurMot(Lettres,
                                                     tDictionnaire);
                  end;
          end;
```

```
    until c = 0;
end.
```

{ ---------- pensez à changer de copie ---------- }

2ème Partie : Récursivité *(5 points)*

2.1 Produit récursif

Voici une fonction itérative qui retourne le produit de deux entiers positifs *n* et *m* :

```
function produitIteratif(n,m:integer):integer;
var
   i : integer;
begin
      produitIteratif := 0;
      for i := 1 to n do
            produitIteratif := produitIteratif + m;
end;
```

a) Identifier dans ce programme itératif un critère d'arrêt et une expression récursive en vue d'écrire une fonction récursive pour faire ce même calcul.

- Critère d'arrêt : n = 1
- produitRecursif := m + produitRecursif (n-1, m)

b) Ecrire une fonction `produitRecursif(n,m)` avec les éléments identifiés en a).

```
function produitRecursif(n,m:integer):integer;
begin
      if n = 1 then
            produitRecursif := m
      else
            produitRecursif := m + produitRecursif(n-1,m);
end;
```

2.2 Fonction d'Ackermann-Péter

La fonction d'Ackermann-Péter est définie comme suit :

$$A(m,n) = \begin{cases} n+1 & si \quad m = 0 \\ A(m-1,1) & si \quad m > 0 \quad et \quad n = 0 \\ A(m-1, A(m,n-1)) & si \quad m > 0 \quad et \quad n > 0 \end{cases}$$

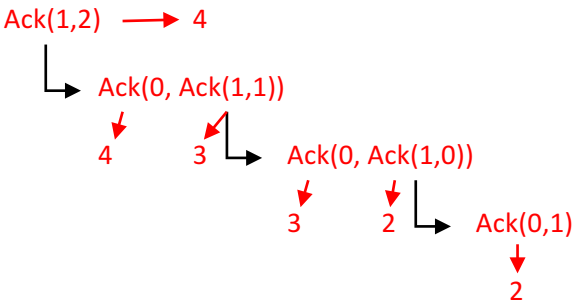
a) Ecrire une fonction récursive `Ack(m,n)` qui retourne la valeur de *A(m,n)* pour deux entiers `m` et `n` positifs ou nuls en entrée.

```
function Ack(m,n : integer) : integer;
begin
      if m = 0 then
            Ack := n + 1
      else if (m >0) and (n = 0) then
```

```
                Ack := Ack(m-1,1)
            else if (m > 0) and (n > 0) then
                Ack := Ack(m-1,Ack(m,n-1));
end;
```

b) Quel est le résultat de l'appel `Ack(1,2)`. Expliciter les appels successifs graphiquement.

Résultat : La sortie Ack(1, 2) est 4. Le dessin ci-dessous montrent les appels puis les retours successifs. Les flèches noirs sont les appels successifs et les flèches rouges les retours de ces appels



{ ---------- pensez à changer de copie ---------- }

3ème Partie : Fichiers *(5 points)*

1. Quel est le point commun principal entre un fichier et un tableau ?
2. Quelle est la différence principale entre un fichier et un tableau ?
3. On souhaite gérer un ensemble de personnes au moyen de fichiers. Une personne est décrite par son nom, son prénom et son âge.
   a. Donner les déclarations (types et variables) pour une personne et pour 3 fichiers de personnes.
   b. Ecrire un programme `Majorite` qui permet de saisir le nom des 3 fichiers, puis de recopier, depuis le 1er fichier, les personnes majeures dans le 2ème fichier et les personnes mineures dans le 3ème fichier. Le programme se terminera en précisant le nombre de personnes présentes dans chaque fichier. *(3 points)*

```
Program Majorite;
type
   personne = record
                    nom,prenom : string;
                    age : integer;
              end;
   fichier= file of personne;
var
   f1,f2,f : fichier;
   pers : personne;
   compteurMin, compteurMaj : integer;
   p, pmin,pmaj : string;

begin
```

```
      writeln('Entrez le nom du ficier des personnes :');
      readln (p);
      writeln('Entrez le nom du ficier des personnes mineures:');
      readln (pmin);
      writeln('Entrez le nom du ficier des personnes mineures:');
      readln (pmaj);
      assign(f,p);
      assign(f1,pmin);
      assign(f2,pmaj);
      reset(f);
      rewrite(f1);
      rewrite(f2);
      compteurMin := 0;
      compteurMaj := 0;
      while not eof(f) do
      begin
            read(f,pers);
            if pers.age < 18 then
               begin
                write(f1,pers);
                compteurMin := compteurMin + 1;
               end
            else
               begin
                write(f2,pers);
                compteurMaj := compteurMaj + 1;
               end;
            read(f,pers);
      end;
       writeln('Il y a : ', compteurMin, 'mineurs');
       writeln('Il y a : ', compteurMaj, 'majeurs');
       writeln('Il y a : ', compteurMin + compteurMaj, 'personnes');
       close(f1);
       close(f2);
       close(f);
end.
```

{ ---------- pensez à changer de copie ---------- }

4ème Partie : procédures, paramètres et variables *(5 points)*

Le programme qui suit permet d'illustrer d'une part la différence entre variables locales et globales et d'autre part le passage de paramètres par variables et par adresses.

a) Expliquer les types des variables utilisées (programme principal, fonction et procédure).

b) Simuler l'exécution du programme et remplir le tableau 1 (dernière page).

c) Modifier les déclarations des paramètres de la fonction `qui` et de la procédure `comment` comme suit :

```
function qui(var d : string ; e : string) : string;
```

```
procedure comment (d : string ; var e : string);
```

Remplir le tableau 2 (dernière page).

d) Commentez les deux tableaux.

```
program Enregistrement;
const c ='mal';
var a, b, d, e : string;

procedure comment (d , e : string);
var a : string;
begin
     a:='c''est';
     e:='pas mal';
     writeln('Comment');
     writeln('a=',a);
     writeln('b=',b);
     writeln('c=',c);
     writeln('d=',d);
     writeln('e=',e);
     writeln('Phrase 1 : ',a,' ',b,' ',c,' ',d);
end;
function qui(D,E:string):string;
var B,C : string;
begin
     b:='va';
     c:='pas';
     d:='bien';
     qui:=e;
     writeln('qui');
     writeln('a=',a);
     writeln('b=',b);
     writeln('c=',c);
     writeln('d=',d);
     writeln('e=',e);
     writeln('Phrase 2 : ',a,' ',b,' ',c,' ',d);
end;

begin
     a:='toto';
     b:='pas';
     e:='mal';
     comment(a,b);
     d:=qui(a,b);
     writeln('Qui_Comment');
     writeln('a=',a);
     writeln('b=',b);
     writeln('c=',c);
     writeln('d=',d);
     writeln('e=',e);
     writeln('Phrase 3 : ',b,' ',c,' ',a,' ',d,' ',e);
end.
```

Page à rendre avec le problème n°4 (procédures, paramètres et variables)

NOM : **Prénom :**

Signature :

Q2 : avant modification
Comment
A= c'est
B= pas
C= mal
D= toto
E= pas mal
Phrase 1- c'est pas mal toto
Qui
A= toto
B= va
C= pas
D= bien
E= pas
Phrase 2 toto va pas bien
Qui_Comment
A= toto
B= pas
C= mal
D= pas
E= mal
Phrase 3- pas mal toto pas mal

Tableau 1.

Q3 : après modification
Comment
A= c'est
B= pas mal
C= mal
D= toto
E= pas mal
Phrase 1- c'est pas mal mal toto
Qui
A= bien
B= va
C= pas
D= bien
E= pas mal
Phrase 2 – bien va pas bien
Qui_Comment
A= bien
B= pas mal
C= mal
D= pas mal
E= mal
Phrase 3- pas mal mal bien pas mal mal

Tableau 2.