

Nom et prénom :

Exercice 1 (3pts) :

Soit la partie déclarative suivante :

Const C=3 ;

Type

Fourniture = (stylo, crayon, gomme, colle, cahier, carnet) ;

Ensemble = 10..30 ;

Var

g, f : fourniture ; e : ensemble ; i, j : integer ;

c1, c2 : char ; s : string ;

Répondre par Faux si l'assertion est fautive et par Vrai sinon:

Instruction	Vrai / faux	Valeur si vrai
g := 'cahier' ;		
f := carnet ;		
write(f) ;		
j := ord(f) * c ;		
i := succ(ord(gomme)) ;		
readln(c1) ;		
c2 := chr(ord(c1)) ;		
e := ord(cahier) * ord(carnet) ;		

Exercice 2 (5pts) :

Soit la grille d'analyse suivante :

DEF FN traitement (T : tab ; n : entier) : entier		
S	LDE	OU
2	Résultat = traitement	
1	Traitement ← x	x
	x = [x ← T [1]] Pour i de 3 à n (pas =2) Faire	i
	Si T[i] > x Alors x ← T[i]	
	Fin Si	
	Fin Pour	
3	Fin traitement	

1) Exécuter cette fonction pour le tableau T

T	12	8	17	5	2	26	17
---	----	---	----	---	---	----	----

2) Quel est le rôle de cette fonction

.....

3) Traduire la fonction en Pascal

.....

Problème (12pts) :

Un laboratoire de recherche scientifique réalise des expériences en biologie. Une expérience est caractérisée par son nom et son résultat (échec, neutre, ou succès). L'informatisation de la gestion des résultats des expériences de ce laboratoire nécessite l'utilisation des deux tableaux suivants :

TE (EXPERIENCE) : est un tableau contenant les noms des expériences. Le nom est une chaîne de caractères non vide et ne dépassant pas 30 caractères.

TR (RESULTAT) : est un tableau qui contient les résultats des différentes expériences codés de la façon suivante :

- -1 : si le résultat est échec
- 0 : si le résultat est neutre
- 1 : si le résultat est succès.

NB : TE[i] et TR[i] contiennent respectivement le nom et le résultat de l'expérience n° i.

Ecrire une Analyse modulaire LABO qui permet de :

- 1) Lire un entier N (longueur des deux tableaux TE et TR) avec $1 \leq N \leq 100$
- 2) Remplir les tableaux TE et TR respectivement par les noms des expériences et le résultat de chaque expérience correspondante.
- 3) Afficher un troisième tableau T contenant les noms des expériences ordonnés selon le résultat obtenu (succès, neutre puis échec). Si plusieurs expériences ont abouti au même résultat, on les place selon l'ordre alphabétique de leur nom.

1) Analyser chacun des modules envisagés dans l'analyse du programme principal

2) Dédire de ce qui précède l'algorithme du programme principal ainsi que les algorithmes des modules envisagés.

Exemple

TE	Bactérie	Cellules	ADN	Protéines	Chlorophylle
----	----------	----------	-----	-----------	--------------

TR	1	0	0	-1	1
----	---	---	---	----	---

T	Bactérie	Chlorophylle	ADN	Cellules	Protéines
---	----------	--------------	-----	----------	-----------

1) Exécuter cette fonction pour le tableau T

T	12	8	17	5	2	26	17
	1	2	3	4	5	6	7

Tableau d'exécution manuelle (fonction Test) :

itérations	Initialisation	1	2	3
Objet				
x	12	17	17	17
i		3	5	7

Retour de la fonction :

Traitement \leftarrow 17

2) Quel est le rôle de cette fonction

Cette fonction permet de déterminer la valeur maximale parmi les éléments d'indice impair.

3) Traduire la fonction en Pascal

```

Function traitement (T : tab ; n : integer) : integer
Var
  x, c, i :integer;

Begin
  x:= T [1];

  for c := 1 to 1 + Round ((n - 3) / 2) do
  begin
    i:= c*2;
    IF T [i+1] > x then  x := T [i];
  End;

  traitement := x;
end;

```

Problème (12pts) :**a. Analyse principale**

Nom : expériences		
S	L.D.E	O.U
4	Résultat = PROC affiche (TE, n)	Affiche, TE, TR, n
3	TE= PROC Tri_TE (TE, TR, n)	Tri_TE
2	TE, TR= PROC Tri (TE, TR, n)	Tri
1	TE, TR = PROC Remplir (TE, TR, n)	Remplir
5	n = PROC saisie (n)	saisie
Fin expériences		

T.D.N.T

Type
Expériences = tableau de 100 chaînes de caractères
Résultats = tableau de 100 entiers

T.D.O globaux

Objet	Type	Rôle
TE	Expériences	Tableau des expériences
TR	Résultats	Tableau des résultats
n	Entier	Nombre de cases
Saisie	Procédure	
Remplir	Procédure	
Tri	Procédure	Permet de trier TE et TR
Tri_bloc	Procédure	
Affiche	Procédure	

Algorithme :

- 0) Début expériences
- 1) PROC saisie (n)
- 2) PROC Remplir (TE, TR, n)
- 3) PROC Tri (TE, TR, n)
- 4) PROC Tri_TE (TE, TR, n)
- 5) PROC affiche (TE, n)
- 6) Fin expériences

b. Analyse de la procédure saisie :

DEF PROC saisie (var n : entier)		
S	L.D.E	O.U
	Résultat = n	
1	n = [] répéter n = donnée ("Donner le nombre de cases de TE et TR : ") jusqu'à (n ≤ 100) et (n ≥ 1)	
2	Fin saisie	

Algorithme :

- 0) DEF PROC **saisie** (var n : entier)
- 1) Répéter
 Ecrire ("Donner le nombre de cases de TE et TR : "), Lire(n)
 Jusqu'à (n ≤ 100) et (n ≥ 1)
- 2) Fin saisie

c. Analyse de la procédure Remplir :

DEF PROC remplir (var TE: expériences; var TR: résultats; n : entier)		
S	L.D.E	O.U
	Résultat = TE, TR	
1	TE, TR = [] pour i de 1 à n faire Répéter TE[i] = donnée ("donner le nom de l'expérience n° ", i " : ") Jusqu'à Long (TE[i]) dans [1..30] Répéter TR[i] = donnée ("donner le résultat de l'expérience n° ", i " : ") Jusqu'à (TR[i] = -1) ou (TR[i] = 0) ou (TR[i] = 1) Fin pour i = compteur	i
2	Fin remplir	

T.D.O Locaux

Objet	Type	Rôle
i	Entier	Compteur

Algorithme :

- 0) DEF PROC remplir (var TE: expériences; var TR: résultats; n : entier)
- 1) Pour i de 1 à n faire
- Répéter
- TE[i] = donnée ("donner le nom de l'expérience n° ", i " : ")
- Jusqu'à Long (TE[i]) dans [1..30]
- Répéter
- TR[i] = donnée ("donner le résultat de l'expérience n° ", i " : ")
- Jusqu'à (TR[i] =-1) ou (TR[i] =0) ou (TR[i] =1)
- Fin pour
- 2) Fin remplir

d. Analyse de la procédure tri (méthode à bulles Ordre décroissant / TR)

DEF PROC tri (var TE : Expériences; var TR : Résultats; n : entier)		
S	L.D.E	O.U
1	Résultat = TE, TR TE, TR = [] Répéter [p ← faux] pour i de 1 à n-1 faire Si TR[i] < TR [i+1] alors p ← vrai permuter1 (TR[i], TR [i+1]) permuter2 (TE[i], TE [i+1]) Fin si Fin pour Jusqu'à p = faux i = compteur	i p Permuter1 Permuter2
2	Fin tri	

T.D.O Locaux

Objet	Type	Rôle
i	Entier	compteur
p	Booléen	Teste des changements
Permuter1	Procédure	Permute 2 entiers
Permuter2	Procédure	Permute 2 chaînes

Algorithme :

0) DEF PROC tri (var TE : Expériences; var TR : Résultats; n : entier)

1) Répéter

 p ← faux

 Pour i de 1 à n-1 faire

 Si TR[i] < TR [i+1] alors p ← vrai

 permuter1 (TR[i], TR [i+1])

 permuter2 (TE[i], TE [i+1])

 Fin si

 Fin pour

Jusqu'à p = faux

2) Fin tri

e. Analyse de la procédure permuter1

DEF PROC permuter1 (var a : entier, var b : entier)		
S	L.D.E	O.U
1	Résultat = a, b	p
2	p ← a	
3	a ← b	
4	b ← p	
4	Fin permuter1	

T.D.O Locaux

Objet	Type	Rôle
P	entier	Variable tampon

Algorithme

0) DEF PROC permuter1 (var a : entier, var b : entier)

1) p ← a

2) a ← b

3) b ← p

4) Fin permuter1

f. Analyse de la procédure permuter2

DEF PROC permuter2 (var a : chaîne, var b : chaîne)		
S	L.D.E	O.U
1	Résultat = a, b	p
2	p ← a	
3	a ← b	
4	b ← p	
4	Fin permuter2	

T.D.O Locaux

Objet	Type	Rôle
P	Chaîne	Variable tampon

Algorithme

- 0) DEF PROC permuter2 (var a : chaîne, var b : chaîne)
- 1) $p \leftarrow a$
- 2) $a \leftarrow b$
- 3) $b \leftarrow p$
- 4) Fin permuter2

g. Analyse de la procédure Tri_TE :

DEF PROC Tri_TE (var TE : Expériences; TR : Résultats; n : entier)		
S	L.D.E	O.U
1	Résultat = TE TE = [k ← 0] Répéter $k \leftarrow k+1$ $J \leftarrow k$ Tant que $(k+1 \leq n)$ et $(TR [k+1] = TR [j])$ faire $k \leftarrow k+1$ Fin tant que Tri_bloc (TE, j, k) Jusqu'à $k = n$ Fin Tri_bloc	k j
2		

T.D.O Locaux

Objet	Type	Rôle
k, j	Entier	

Algorithme :

0) DEF PROC Tri_TE (var TE : Expériences; TR : Résultats; n : entier)

1) $k \leftarrow 0$

Répéter

$k \leftarrow k+1$

$J \leftarrow k$

Tant que $(k+1 \leq n)$ et $(TR[k+1] = TR[j])$ faire

$k \leftarrow k+1$

Fin tant que

Tri_bloc (TE, j, k)

Jusqu'à $k = n$

2) Fin Tri_bloc

h. Analyse de la procédure Tri_bloc (méthode par insertion) :

DEF PROC Tri_bloc (var TE : Expériences; j, k : entier)		
S	L.D.E	O.U
	Résultat = TE	
1	TE = [] Pour x de j+1 à k faire $tmp \leftarrow TE[x]$ $y \leftarrow x$ Tant que $(y - 1 \geq j)$ et $(TE[y - 1] > tmp)$ faire $TE[y] \leftarrow TE[y-1]$ $y \leftarrow y - 1$ Fin Tant que $TE[y] \leftarrow tmp$ Fin pour x, y = compteur	x y tmp
2	Fin Tri_bloc	

T.D.O Locaux

Objet	Type	Rôle
x, y	Entier	Compteur
tmp	Chaîne	

Algorithme :

- ```

0) DEF PROC Tri_bloc (var TE : Expériences; j, k : entier)
1) Pour x de j+1 à k faire
 tmp ← TE[x]
 y ← x
 Tant que (y - 1 ≥ j) et (TE[y - 1] > tmp) faire
 TE[y] ← TE[y-1]
 y ← y - 1
 Fin Tant que
 TE[y] ← tmp
Fin pour
2) Fin Tri_bloc

```

**i. Analyse de la procédure affiche :**

| DEF PROC affiche (TE : Expériences; n : entier) |                                                                                                             |     |
|-------------------------------------------------|-------------------------------------------------------------------------------------------------------------|-----|
| S                                               | L.D.E                                                                                                       | O.U |
| 1                                               | Résultat = affichage<br>affichage = [ ] Pour i de 1 à n faire<br>écrire (TE[i])<br>Fin pour<br>i = compteur | i   |
| 2                                               | Fin affiche                                                                                                 |     |

**T.D.O Locaux**

| Objet | Type   | Rôle     |
|-------|--------|----------|
| i     | Entier | Compteur |

**Algorithme :**

- ```

0) DEF PROC affiche (N: Nom; M : moyenne; k : entier)
1) i ← 1
    Tant que (i ≤ k) ET (M[i] ≥ 10) faire
        écrire (N[i])
        i ← i + 1
    Fin tant que
    Si i = 1 alors écrire ("Aucun élève admis !")
    Fin si
2) Fin affiche

```

Barème

	Points
Analyse principale	2
Types	0,5
TDO globaux	0,5
Saisie(n)	1
Remplir_TE ()	1,25
Remplir_TR ()	1,25
Tri1 ()	1,75
Tri2 ()	1,5
Affichage	0,75
Algorithmes	1,5

```
program resultats_experiences;
uses wincrt;
Type
  experiences=array[1..10]of string;
  resultats=array[1..10]of integer;
var
  TE:experiences;
  TR:resultats;
  n:integer;

procedure saisie(var k:integer);
begin
  repeat
    Write('Donner le nombre de cases de TE et TR :');
    readln(k);
  until k in [1..100];
end;

{      **      }
procedure remplir (var TE:experiences; var TR:resultats; n:integer);
var
  i:integer;
begin
  for i:=1 to n do
    begin
      repeat
        write('Donner le nom de l'expérience n° ',i,' : ');
        readln(TE[i]);
      until length(TE[i]) in [1..30];
      repeat
        write('Donner le résultat de l'expérience n° ',i,' : ');
        readln(TR[i]);
      until (TR[i]= -1) or (TR[i]= 0) or (TR[i]= 1);
      writeln;
    end;
  end;

{      **      }
Procedure permuter1( var a:integer; var b:integer);
Var
  p : integer;
begin
  p:=a;
  a:=b;
  b:=p;
end;

Procedure permuter2( var a:string; var b:string);
Var
  p : string;
begin
  p:=a;
  a:=b;
  b:=p;
end;

{      méthode : tri à bulles      }

Procedure tri(var TE:experiences;var TR:resultats; n:integer);
var
  i:integer;
  p:boolean;
```

```

begin
  repeat
    p:=false;
    for i:=1 to n-1 do
      begin
        if TR[i]<TR[i+1] then
          begin
            p:=true;
            permuter1(TR[i],TR[i+1]);
            permuter2(TE[i],TE[i+1]);
          end;
        end;
      until p = false;
    end;
  {          **          }
  procedure tri_bloc(var TE:experiences;j,k:integer);
  var
    x,y:integer;
    tmp:string;
  begin
    for x:=j+1 to k do
      begin
        tmp:=tE[x];
        y:=x;
        while (y-1>=j) and (tE[y-1]>tmp) do
          begin
            tE[y]:=tE[y-1];
            dec(y,1);
          end;
        tE[y]:=tmp;
      end;
    end;
  {          **          }
  procedure tri_TE(var TE:experiences; TR:resultats; n:integer);
  var j,k:integer;
  begin
    k:=0;
    repeat
      k:=k+1;
      j:=k;

      while (k+1 <=n) and (TR[k+1]= TR[j]) do
        inc(k,1);

      tri_bloc(TE,j,k);
      {démonstration des limites}
      writeln('j : ',j,' k : ',k);
    until k=n;
  end;

  procedure affiche(TE:experiences;n:integer);
  var i:integer;
  begin
    writeln('Le tableau des expériences est : ');
    for i:=1 to n do
      write(TE[i]:5);
    end;
  {          **          }
  begin
    saisie(n);
    remplir(TE,TR,n);
    tri(TE,TR,n);
    tri_TE(TE,TR,n);
    affiche(TE,n);
  end.

```

{Correction DS n°3 07-08}

{Problème 12 pts, version 2 une seule procédure de Tri qui organise simultanément TR et TE
Par résultats et par nom d'expériences.}

```
program resultats_experiences;
uses wincrt;
Type
  experiences=array[1..10]of string;
  resultats=array[1..10]of integer;
var
  TE:experiences;
  TR:resultats;
  n:integer;

procedure saisie(var k:integer);
begin
  repeat
    Write('Donner le nombre de cases de TE et TR :');
    readln(k);
  until k in [1..100];
end;

{      **      }
procedure remplir (var TE:experiences; var TR:resultats; n:integer);
var
  i:integer;
begin
  for i:=1 to n do
    begin
      repeat
        write('Donner le nom de l'expérience n° ',i,' : ');
        readln(TE[i]);
      until length(TE[i]) in [1..30];
      repeat
        write('Donner le résultat de l'expérience n° ',i,' : ');
        readln(TR[i]);
      until (TR[i]= -1) or (TR[i]= 0) or (TR[i]= 1);
      writeln;
    end;
end;

{      **      }
Procedure permuter1( var a:integer; var b:integer);
Var
  p : integer;
begin
  p:=a;
  a:=b;
  b:=p;
end;

Procedure permuter2( var a:string; var b:string);
Var
  p : string;
begin
  p:=a;
  a:=b;
  b:=p;
end;
```

Version 2.
Tri simultané

```
{           méthode : tri à bulles           }
```

```
Procedure tri(var TE:experiences;var TR:resultats; n:integer);
var
  i:integer;
  p:boolean;
begin
  repeat
    p:=false;
    for i:=1 to n-1 do
      begin
        if TR[i]<TR[i+1] then
          begin
            p:=true;
            permuter1(TR[i],TR[i+1]);
            permuter2(TE[i],TE[i+1]);
          end
        Else if TR[i]=TR[i+1] then if TE[i]>TE[i+1] then
          begin
            p:=true;
            permuter2(TE[i],TE[i+1]);
          end;
      end;
    until p = false;
end;
{           **           }
```

```
procedure affiche(TE:experiences;n:integer);
var
  i:integer;
begin
  writeln('Le tableau des expériences est : ');
  for i:=1 to n do
    write(TE[i]:5);
end;
{           **           }
```

```
begin
  saisie(n);
  remplir(TE,TR,n);
  tri(TE,TR,n);
  affiche(TE,n);
end.
```

Version 2.
Tri simultané