Le Commissariat Régional de l'Education de Gabés

Centre Régional de l'Education et de la Formation Continue de Gabès

Matière : Informatique

Une correction des épreuves pratiques pour les sections scientifiques

[*Session Mai 2015*]

Réalisé par :

- **Les inspecteurs :** **M. Hédi HANNACHI & M. Ali Essghaier DELHOUMI**
- **Les enseignants d'Informatique aux lycées du C.R.E. de Gabes**

Année – Scolaire : 2015/2016

Séance1-Sujet1

Pour sécuriser l'envoi des messages, deux chercheurs cryptent leurs messages en utilisant le principe suivant :

1. Saisir le message à crypter **msg**, sachant qu'il est composé uniquement par des lettres,
2. Remplir un tableau **T** par les ordres alphabétiques des lettres de **msg** de façon à ce que **T[i]** lui correspond l'ordre de **msg[i]** (Sachant que "A" et "a" sont d'ordre l, "B" et "b" sont d'ordre 2, …),
3. Remplacer chaque **T[i]** par **$(T[i])^e$ mod (p*q)** avec **p, q** et **e** trois constantes ayant pour valeurs respectivement **17, 19** et **5**.

Le tableau **T** ainsi obtenu représente le code de la chaine **msg**.

Exemple :

Pour la chaîne **msg**="Bonjour", **T** sera remplit initialement comme suit :

T	2	15	14	10	15	21	18
	1	2	3	4	5	6	7

En effet "B" et d'ordre alphabétique 2, "o" est d'ordre alphabétique 15, …

Après avoir coder en remplaçant chaque **T[i]** par **$(T[i])^e$ mod (p*q) on obtient :**

T	32	2	29	193	2	89	18
	1	2	3	4	5	6	7

En effet :
T[1] est remplacé par $(T[1])^e \bmod (p*q)=2^5 \bmod (17*19) =32$
T[2] est remplacé par $(T[2])^e \bmod (p*q)=15^5 \bmod (17*19)=2$
Etc.

Travail demandé :

Ecrire un programme Pascal qui permet de saisir une chaîne non vide formée uniquement par des lettres, de la crypter selon le principe décrit ci-dessus et d'afficher le tableau de code obtenu.

Analyse du programme principal

```
❹ Résultat = [ ] Pour i de 1 à Long(Msg[i]) Faire
                 Ecrire(Tf[i])
             Fin Pour
❶Msg = [ ]
             Répéter
                 Ecrire("Message : "), Lire(Msg)
             Jusqu'à (Fn Alpha(Msg)=Vrai) et (long(Msg)>0)
❸Tf= [   ]
             Pour i de 1 à Long(Msg) Faire
                 Tf[i] ← Fn Puiss(T[i], e) mod (p*q)
             Fin Pour
❷T =[   ]
             Pour i de 1 à Long(Msg) Faire
                 T[i] ← Ord(Majus(Msg[i]))-64
             Fin Pour
```

Le Tableau de Déclaration des Nouveaux Types

Type
Bac = Tableau de 255 entiers

Le tableau de déclaration des objets globaux

Objet	Type / Nature	Rôle
I	Octet	Compteur
Msg	Chaîne	Le message à crypter
T	Bac	Le tableau contenant le ***Résultat*** du cryptage
Alpha	Fonction	Qui vérifie si une chaine est alphabétique ou non
Puiss	Fonction	Qui retourne la puissance d'ordre K d'un entier X
E	Constante de valeur = 5	
P	Constante de valeur = 17	
Q	Constante de valeur = 19	

Analyse de la fonction Alpha

Def Fn Alpha(X :Chaîne) : Booléen

Résultat = Alpha
❸ Alpha ←Ok
❷ Ok ← Majus(X[i]) dans ["A".."Z"]
❶ I [i←0]
Répéter
I←i+1
Jusqu'à (Non(Majus(X[i]) dans ["A".."Z"]) ou (i=Long(x))

Le tableau de déclaration des objets locaux

Objet	Type / Nature	Rôle
I	Octet	Compteur
Ok	Booléen	Aura comme valeur **Faux** si le dernier caractère testé n'est pas une lettre ou **Vrai** dans le cas contraire

Analyse de la fonction Puiss

Def Fn Puiss(X ,K:Octet) : Entier long

Résultat = Puiss
❷ Puiss ←P
❶ P =[P←1]
Pour i de 1 à K Faire
P←P*X
Fin Pour

Le tableau de déclaration des objets locaux

Objet	Type / Nature	Rôle
I	Octet	Compteur
P	Entier Long	Variable locale qui devra comporter le ***Résultat*** à retourner à la fonction

La traduction Pascal:

```
Program Cryptage1;
uses wincrt;
const e=5;p=17; q=19;
```

```
type tab=array[1..50] of integer;
var t:tab;msg:string[50]; i:byte;
                    {******* la fonction Alpha *******}
function Alpha(x:string):boolean;
var i:byte; Ok:boolean;
begin
i:=0;
repeat
i:=i+1;
until (not(upcase(x[i]) in ['A'..'Z'])) or (i=length(x));
Ok:= upcase(x[i]) in ['A'..'Z'];
alpha := Ok;
end;
                    {******* la fonction Puiss *******}
function Puiss(x,k:byte):longint;
var i:byte;p:longint;
begin
p:=1;
for i:=1 to k do p:=p*x;
puiss:=p;
end;
                    {******* le programme principal *******}
begin
repeat
write('message = '); readln(msg);
until (alpha(msg)) and (length(msg)>0);
for i:=1 to length(msg) do t[i]:= ord(upcase(msg[i]))-64;
for i := 1 to length(msg) do t[i]:=puiss(t[i],e) mod (p*q);
for i := 1 to length(msg) do write(t[i]:5);
end.
```

Séance1-Sujet2

Pour sécuriser l'envoi des messages, deux chercheurs cryptent leurs messages en utilisant une clé de cryptage selon le principe suivant :

1. Saisir le message à crypter **msg**, sachant qu'il est composé par des lettres majuscules et des espaces.
2. Saisir la clé de cryptage qui est une chaîne de caractères **chcle** composée uniquement par des chiffres et ayant la même longueur que le message à crypter.
3. Remplacer chaque lettre du message **msg**, d'ordre alphabétique **i**, par la lettre d'ordre alphabétique **j** avec **j=i+c**, sachant que **c** est le chiffre de la chaîne **chcle** ayant le même indice que la lettre à crypter.

N.B.

- L'espace ne sera pas crypté,
- Si **j** dépasse 26, on reprend les lettres alphabétiques dès le début.

Exemple :

Soit le message "**EXCELLENTE PERFORMANCE**" et soit la clé "**19546327384016536284 51**"

Message initial :	E	X	C	E	L	L	E	N	T	E		P	E	R	F	O	R	M	A	N	C	E
La clé de cryptage :	1	9	5	4	6	3	2	7	3	8	4	0	1	6	5	3	6	2	8	4	5	1
Message codé :	F	G	H	I	R	O	G	U	W	M		P	F	X	K	R	X	O	I	R	H	F

En effet :

- La lettre "E" est d'ordre alphabétique 5, elle sera remplacée par la lettre d'ordre alphabétique 5+1=6 c'est-à-dire "F"
- La lettre "X" est d'ordre alphabétique 24, elle sera remplacée par la lettre d'ordre alphabétique 24+9=33, 33 MOD 26 7 c'est-à-dire "Gᵗ'
- etc.

Travail demandé :

Ecrire un programme Pascal qui permet de saisir un message et une clé de cryptage en respectant les contraintes citées ci-dessus, puis d'afficher le message crypté en utilisant le principe décrit précédemment.

Analyse du programme principal

❹ ***Résultat*** = Ecrire ("Le message codé est :", Msgf)
❸ Msgf = Proc Cryptage(Cle, Msg)
❷ Cle= [ ] Répéter
Ecrire("Clé = "), Lire(Cle)
Jusqu'à (Fn Verif2(Cle)=Vrai) et (Long(Cle)=Long(Msg))
❶ Msg = [ ] Répéter
Ecrire("Message = "), Lire(Msg)
Jusqu'à (Fn Verif1(Msg)=Vrai) et (Long(Msg)>0)

Le tableau de déclaration des objets globaux

Objet	Type / Nature	Rôle
Msg	Chaine	
Cryptage	Procédure	
Cle	Chaîne	
Verif2	Fonction	
Verif1	Fonction	

Analyse de la fonction Verif1

Def Fn Verif1(Ch :Chaîne) : Booléen

Résultat = Verif1
❸ Verif1 ←Test
❷ Test ← Ch[i] dans ["A".."Z","␣"]
❶ I [i←0]
Répéter
I←i+1
Jusqu'à (Non(Ch[i] dans ["A".."Z","␣"])) ou (i=Long(ch))

Le tableau de déclaration des objets locaux

Objet	Type / Nature	Rôle
I	Octet	Compteur
Test	Booléen	Aura comme valeur **Faux** si le dernier caractère testé n'est ni lettre majuscule, ni espace "␣" ou **Vrai** dans le cas contraire

Analyse de la fonction Verif2

Def Fn Verif2(Ch :Chaîne) : Booléen

Résultat = Verif2
❸ Verif2 ←Test

❷ Test ← Ch[i] dans ["0".."9"]
❶ I [i←0]
Répéter
I←i+1
Jusqu'à (Non(Ch[i] dans ["0".."9"])) ou (i=Long(ch))

Le tableau de déclaration des objets locaux

Objet	Type / Nature	Rôle
I	Octet	Compteur
Test	Booléen	Aura comme valeur **Faux** si le dernier caractère testé n'est pas un chiffre ou **Vrai** dans le cas contraire

Analyse de la procédure Cryptage

Def Proc Cryptage (Cle: Chaîne; Var Msg: Chaîne)

Résultat = Msg
❶ Msg = [ ]
Pour i de 1 à Long(Msg) Faire
Si Msg[i] ≠ "␣" Alors
x← Ord(Msg[i])-64
J← (x +ord(Cle[i])-48) mod 26
Si j=0 Alors Msg[i]← "Z"
Sinon Msg[i]←Chr(j+64)
Fin Si
Fin Si
Fin Pour

Le tableau de déclaration des objets locaux

Objet	Type / Nature	Rôle
I	Octet	Compteur
J	Octet	Le nouvel numéro d'ordre demandé par la règle de cryptage
X	Octet	Le numéro d'ordre d'une lettre majuscule dans l'alphabet

La traduction Pascal:

```
Program Cryptage2;
uses wincrt;
var Msg,Cle:string;
                    {******* la fonction Verif1 *******}
function verif1(ch:string):boolean;
var test:boolean; i:Byte;
begin
i:=0;
repeat
i:=i+1;
until (not(ch[i] in ['A'..'Z',' '])) or (i=length(ch));
test:=ch[i] in ['A'..'Z',' '];
verif1:=test;
end;
                    {******* la fonction Verif2 *******}
function verif2(ch:string):boolean;
```

```
var test:boolean; i:Byte;
begin
i:=0;
repeat
i:=i+1;
until (not(ch[i] in ['0'..'9'])) or (i=length(ch));
test:=ch[i] in ['0'..'9'];
verif2:=test;
end;
                    {******* la procédure cryptage *******}
procedure Cryptage( Cle:string;Var Msg :string);
var i,j,x:Byte;
begin
for i:=1 to length(Msg) do
if Msg[i]<>' ' then
        begin
                x:=ord(Msg[i])-64;
                j:=(x+ord(Cle[i])-48) mod 26;
                if j=0 then Msg[i]:='Z' else Msg[i]:=chr(j+64);
        end;
end;
                    {******* le programme principal *******}
begin
repeat
write('Message  = '); readln(Msg);
until (verif1(Msg)=true)and(length(Msg)>0);
repeat
write('Clé = '); readln(Cle);
until (verif2(Cle)) and (length(Msg)=length(Cle));
cryptage(Cle,Msg);
write('Message codé = ',Msg);
end.
```

Séance1-Sujet3

Pour sécuriser l'envoi des messages, deux chercheurs cryptent leurs messages en utilisant une clé de cryptage selon le principe suivant :

1. Saisir le message à crypter **msg**, sachant qu'il est composé par des lettres minuscules et des espaces,
2. Saisir une clé de cryptage **chcle** qui est une chaîne formée uniquement par des lettres minuscules et ayant la même longueur que le message à crypter,
3. Remplacer chaque lettre du message **msg** d'indice **i** par la lettre minuscule d'ordre alphabétique **k** sachant que:
   - k = ABS (ord(msg[i]) – ord(chcle[i]))+ 1
   - L'espace ne sera pas crypté.

Exemple : soit le message suivant : "**bonne␣ reception**" et soit la clé "**homeofhappiness**"

Message :	**b**	**o**	**n**	**n**	**e**	**␣**	**r**	**e**	**c**	**e**	**p**	**t**	**i**	**o**	**n**
La clé de cryptage :	**h**	**o**	**m**	**e**	**o**	**f**	**h**	**a**	**p**	**p**	**i**	**n**	**e**	**s**	**s**
Message crypté :	**g**	**a**	**b**	**j**	**k**	**␣**	**k**	**e**	**n**	**l**	**h**	**g**	**e**	**e**	**f**

En effet :

- La lettre "b" sera remplacée par la lettre d'ordre alphabétique k =ABS(ord("b") - ord("h")) + 1 qui est "g" En effet, k = ABS (66-72)+1 = 7 qui est l'ordre alphabétique de la lettre "g".
- La lettre "o" sera remplacée par la lettre d'ordre alphabétique k =ABS(ord("o")-ord("o")) + 1 qui est "a". En effet, k = ABS (79-79) +1 = 1 qui est l'ordre alphabétique de la lettre "a" .
- etc.

Travail demandé :

Ecrire un programme Pascal qui permet de saisir un message **msg** et une clé de cryptage **chcle** en respectant les contraintes citées ci-dessus puis d'afficher le message crypté en utilisant le principe décrit précédemment.

Analyse du programme principal

❹ ***Résultat*** = Ecrire ("Le message crypté est :", Msgf)
❸ Msgf = Proc Cryptage(Chcle, Msg)
❷ Chcle= [ ] Répéter
Ecrire("Clé = "), Lire(Chcle)
Jusqu'à (Fn Verif(Chcle)=Vrai) et (Long(Chcle)=Long(Msg)) et (pos("␣",Chcle)=0)
❶ Msg = [ ] Répéter
Ecrire("Message = "), Lire(Msg)
Jusqu'à Fn Verif(Msg)=Vrai

Le tableau de déclaration des objets globaux

Objet	Type / Nature	Rôle
Msg	Chaine	
Cryptage	Procédure	
Chcle	Chaîne	
Verif	Fonction	

Analyse de la fonction Verif

Def Fn Verif(X :Chaîne) : Booléen

Résultat = Verif
❸ Verif ←Test
❷ Test ← X[i] dans ["a".."z","␣"]
❶ I [i←0]
Répéter
I←i+1
Jusqu'à (Non(X[i] dans ["a".."z","␣"])) ou (i=Long(X))

Le tableau de déclaration des objets locaux

Objet	Type / Nature	Rôle
I	Octet	Compteur
Test	Booléen	Aura comme valeur **Faux** si le dernier caractère testé n'est ni lettre minuscule, ni espace "␣" ou **Vrai** dans le cas contraire

Analyse de la procédure Cryptage

Def Proc Cryptage (Chcle: Chaîne; Var Msg: Chaîne)

Résultat = Msg

❶ Msg = [ ]
Pour i de 1 à Long(Msg) Faire
Si Msg[i] ≠ "␣" Alors
k← abs(Ord(Msg[i])-ord(Chcle[i]))+1
Msg[i]←Chr(k+96)
Fin Si
Fin Pour

Le tableau de déclaration des objets locaux

Objet	Type / Nature	Rôle
I	Octet	Compteur
K	Octet	Le nouvel numéro d'ordre demandé par la règle de cryptage

La traduction Pascal:

```
program Cryptage3;
uses wincrt;
var msg,chcle:string[50];
                    {******* la fonction Verif *******}
function verif (x:string):boolean;
var i:byte;
test:boolean;
begin
i:=0;
repeat
i:=i+1;
until (not(x[i]in['a'..'z',' '])) or(i=length(x));
test:=x[i]in['a'..'z',' '];
verif:=test;end;
                    {******* la procédure Cryptage *******}
Procedure Cryptage(Chcle:string; Var Msg:string);
var  i,k:byte;
begin
        for i:= 1 to length(msg) do
        if msg[i]<>' ' then begin
                        k:=abs(ord(msg[i])-ord(chcle[i]))+1;
                        msg[i]:= chr(96+k);
                        end;
end;
                    {******* le programme principal *******}
begin
repeat
write('Message = '); readln(msg);
until verif(msg)=true;
repeat
write('Chcle = ');readln(chcle);
until (verif(chcle)) and (pos(' ',chcle)=0)and (length(chcle)=length(msg));
Cryptage(Chcle,Msg);
write('Le message crypté est : ',Msg);
end.
```

Séance1-Sujet4

Pour sécuriser l'envoi des messages, deux chercheurs cryptent leurs messages en utilisant le principe suivant :

l. Saisir le message à crypter **msg**, sachant qu'il est composé par des lettres et des espaces,

2. Faire la somme des chiffres du code ASCII de chaque lettre du message **msg**. Dans le cas où cette somme n'est pas un nombre à un seul chiffre on reprend l'addition jusqu'à obtenir un seul chiffre auquel on ajoute une valeur aléatoire allant de 0 à 17. Le nombre obtenu représentera le rang alphabétique de la lettre de remplacement en majuscule.

N.B : L'espace ne sera pas crypté.

Exemple : Pour le message "**Bac␣Sc**", on aura après cryptage le ***Résultat*** suivant :"**RSL␣RZ**". En effet :

- La lettre "B" est remplacée par la lettre "R" car le code ASCII de "B" est 66 et après addition des chiffres on obtient 3 (6+6=12➔1+2=3) et si la valeur aléatoire est 15, l'ordre alphabétique du caractère de remplacement est 18=3+15 qui est "R"
- La lettre "a" est remplacée par la lettre "S" car le code ASCII de "a" est 97 et après addition des chiffres on obtient 7 (9+7=16+1+6=7) et si la valeur aléatoire est 12, l'ordre alphabétique du caractère de remplacement est 19=7+12 qui est "S"
- La lettre "c" est remplacée par la lettre "L" car le code ASCII de "c" est 99 et après addition des chiffres on obtient 9 (9+9=18 ➔ 1+8=9) et si la valeur aléatoire est 3, l'ordre alphabétique du caractère de remplacement est 12=3+9 qui est "L"
- La lettre "S" est remplacée par la lettre "R" car le code ASCII de "S" est 83 et après addition des chiffres on obtient 2 (8+3=11➔1+1=2) et si la valeur aléatoire est 16, l'ordre alphabétique du caractère de remplacement est 18=2+16 qui est "R"
- La lettre "c^{t}" est remplacée par la lettre "Z' car le code ASCII de "c" est 99 et après addition des chiffres on obtient 9 (9+9=18➔1+8=9) et si la valeur aléatoire est 17, l'ordre alphabétique du caractère de remplacement est 26=9+17 qui est "Z"

Travail demandé :

Ecrire un programme Pascal qui permet de saisir une chaîne non vide formée par des lettres et des espaces, de la crypter selon le principe décrit ci-dessus et d'afficher le ***Résultat*** obtenu.

Analyse du programme principal

❸ ***Résultat*** = Ecrire ("Le message crypté est :", Msgf)
❷ Msgf = Proc Cryptage(Msg)
❶ Msg = [ ] Répéter
Ecrire("Message = "), Lire(Msg)
Jusqu'à Fn Verif(Msg)=Vrai

Le tableau de déclaration des objets globaux

Objet	Type / Nature	Rôle
Msg	Chaine	
Cryptage	Procédure	
Verif	Fonction	

Analyse de la fonction Verif

Def Fn Verif(X :Chaîne) : Booléen

Résultat = Verif
❸ Verif ←Test
❷ Test ← X[i] dans ["a".."z","␣","A".."Z"]
❶ I [i←0]
Répéter
I←i+1
Jusqu'à (Non(X[i] dans ["a".."z","␣","A".."Z"])) ou (i=Long(X))

Le tableau de déclaration des objets locaux

Objet	Type / Nature	Rôle
I	Octet	Compteur
Test	Booléen	Aura comme valeur **Faux** si le dernier caractère testé n'est ni lettre minuscule, ni espace "␣" ou **Vrai** dans le cas contraire

Analyse de la procédure Cryptage

Def Proc Cryptage (Var Msg: Chaîne)

Résultat = Msg
❶ Msg = [ ]
Pour i de 1 à Long(Msg) Faire
Si Msg[i] ≠ "␣" Alors
X← Ord(Msg[i])
X← X div 100+X mod 100 div 10+ X mod 10
X← X div 10 + X mod 10
Msg[i]←Chr(64+X+Aléa(18))
Fin Si
Fin Pour

Le tableau de déclaration des objets locaux

Objet	Type / Nature	Rôle
I	Octet	Compteur
X	Octet	Le nouvel numéro d'ordre demandé par la règle de cryptage

La traduction Pascal:

```
Program Cryptage4;
uses wincrt;
var msg:string[50];
                    {******* la fonction Verif *******}
function verif(x:string):boolean;
var i:byte;
begin
i:=0;
repeat
i:=i+1;
until (not(x[i] in ['A'..'Z','a'..'z',' '])) or(i=length(x));
verif:=x[i] in ['A'..'Z','a'..'z',' ']; end;
                    {******* la procédure Cryptage *******}
```

```
Procedure Cryptage(Var Msg:String);
var i,x:Byte;
begin
for i :=1 to length(msg) do
if msg[i]<>' ' then begin
      x:=ord(msg[i]);
      x:=x div 100 +x mod 100 div 10+ x mod 10;
       x:= x div 10 + x mod 10;
      msg[i]:=chr(64+x + random(18));
      end;
end;
                        {******* le programme principal *******}
begin
randomize;
repeat
write('Message = '); readln(msg);
until verif (msg);
Cryptage(Msg);
write('Le message crypté est : ', Msg);
end.
```

Séance2-Sujet1

Un hôtel souhaite attribuer des séjours gratuits à ses résidents à l'occasion de la fête de fin d'année en se basant sur leurs numéros de réservation qui sont des entiers de 4 chiffres.
Les résidents gagnants sont ceux qui possèdent plus de nombres premiers formés à partir de leurs numéros de réservation (le nombre lui-même, les nombres formés de trois chiffres adjacents, les nombres formés de deux chiffres adjacents et les nombres formés par un seul chiffre).
Exemple
Pour les numéros de réservation suivants:

3322	4774	3114	1012	3577	2291	1854	3149	4766	1579
1	2	3	4	5	6	7	8	9	10

Les numéros de réservation des résidents gagnants sont : 3577 et 1579 puisque :

- 3577 possède 5 nombres premiers qui sont 3, 5, 7, 7 et 577
- 1579 possède 5 nombres premiers qui sont 5, 7, 79, 157 et 1579

N.B. : Un nombre est dit premier s'il n'est divisible que par 1 et par lui-même. Par définition, 1 n'est pas premier.

Travail demandé

Ecrire un programme Pascal qui permet de remplir un tableau **T** par **N** (**10 ≤N ≤100**) numéros de réservation, puis d'afficher la liste des résidents gagnants.

Analyse du programme principal

❸ ***Résultat*** = Ecrire ("Les numéros de réservation gagnants sont: ")
Pour i de 1 à N Faire
Si Fn Nombre(T[i])= Fn Max(T,N) Alors Ecrire(t[i])
Fin Si
Fin Pour

❶ N = [ ]
Répéter
Ecrire("N = "), Lire(N)
Jusqu'à N dans [10..100]
❷T =[ ]
Pour i de 1 à N Faire
Répéter
Ecrire("T[",i,"]=")
Lire(T[i])
Jusqu'à (T[i]>999) et (T[i]<10000)
Fin pour

Le Tableau de Déclaration des Nouveaux Types

Type
Bac = Tableau de 100 entiers

Le tableau de déclaration des objets globaux

Objet	Type / Nature	Rôle
I	Octet	
N	Octet	
T	Bac	
Nombre	Fonction	
Max	Fonction	

Analyse de la fonction Nombre

Def Fn Nombre(K :Entier) : Octet

Résultat = Nombre
❷ Nombre ←Nb
❶ Nb = [Nb←0, Convch(K, Ch)]
Pour i de 1 à Long(Ch) Faire
Pour j de 1 à Long(Ch)-i+1 Faire
Valeur(Souschaîne (Ch, j, i), N, er)
Si Fn Premier(N) = Vrai Alors Nb←Nb+1
Fin Si
Fin Pour
Fin Pour

Le tableau de déclaration des objets locaux

Objet	Type / Nature	Rôle
Nb	Octet	
N	Octet	
I	Octet	
J	Octet	
Er	Entier	
Ch	Chaine[4]	
Premier	Fonction	

Analyse de la fonction Max

Def Fn Max(T : Bac ; N :Octet) : Octet

Résultat = Max
❷ Max🡐 M
❶M=[M🡐 Fn Nombre(T[1])]
Pour i de 2 à N Faire
Si Fn Nombre(T[i])>M Alors M🡐 Fn Nombre(T[i])
Fin Si
Fin Pour

Le tableau de déclaration des objets locaux

Objet	Type / Nature	Rôle
M	Octet	
I	Octet	
Nombre	Fonction	

Analyse de la fonction Premier

Def Fn Premier (K :Entier) :Booléen

Résultat = Premier
❷ Premier 🡐 test
❶ test = [ ]
Si K<2 Alors test 🡐 Faux
Sinon si K=2 Alors test 🡐 Vrai
Sinon si K mod 2 = 0 Alors test 🡐 Faux
Sinon
i🡐1
Répéter
i🡐i+2
Jusqu'à (i>Racinecarré(K)) ou (K mod i =0)
Test 🡐 i>Racinecarré(K)
Fin Si

Le tableau de déclaration des objets locaux

Objet	Type / Nature	Rôle
i	Octet	
Test	Fonction	

La traduction Pascal:

```
program gagnants ;
uses wincrt;
type bac=array[1..100] of integer;
var t:bac;  i,n:byte;
                        {******* la fonction premier *******}
function premier(k:integer):boolean;
var i:integer;    test:boolean;
begin
if k<2 then test:=false
      else if k=2 then test:=true
      else if k mod 2 = 0 then test:=false
```

```
                else
                begin
                i:=1;
                repeat
                i:=i+2
                until (i>sqrt(k)) or (k mod i =0);
                test:= i>sqrt(k);
                end;
premier:=test;
end;
                        {******* la fonction nombre  *******}
function nombre(k:integer):byte;
var nb,i,j:byte; n,er: integer;     ch:string[4];
begin
nb:=0;
str(k,ch);
for i :=1 to length(ch) do
        for j:=1 to length(ch)-i+1 do
                begin
                val(copy(ch,j,i),n,er);
                if premier(n) then nb:=nb+1;
                end;
nombre:=nb;
end;
                        {******* la fonction max *******}
function max ( t:bac ; n:byte ):byte ;
var i,m:byte ;
begin
m:= nombre(t[1]) ;
for i:=2 to n do
if nombre(t[i])>m then m:=nombre(t[i]) ;
max:=m;
end ;
                        {******* le programme principal *******}
begin
repeat
write('n = '); readln(n);
until n in [10..100];
for i:=1 to n do
repeat
write('t[',i,']= '); readln(t[i]);
until (t[i]>999) and (t[i]<10000);
write ('les numéros de réservation des résidents gagnants sont : ' ) ;
for i:=1 to n do
if nombre(t[i]) = max(t,n) then write(t[i]:5);
end .
```

Séance2-Sujet2

Un nombre **X** de trois chiffres est dit premier circulaire s'il est premier et que tous les nombres formés par les combinaisons de ses trois chiffres ainsi que tous ceux formés par les combinaisons de deux chiffres sont aussi premiers.
Exemple :

X=311 est un nombre premier circulaire car **311, 131, 113, 11, 31, 13** sont premiers.

Travail demandé :

Ecrire un programme Pascal qui permet de remplir un tableau **T** de **N (5 ≤ N ≤ 30)** entiers positifs de trois chiffres, de chercher et d'afficher tous les entiers premiers circulaires de **T**.

Analyse du programme principal

❸ ***Résultat*** = [ ] Pour i de 1 à N Faire
Si Fn Circulaire(T[i]) Alors Ecrire(T[i])
Fin Si
Fin Pour
❶ N = [ ]
Répéter
Ecrire("N = "), Lire(N)
Jusqu'à N dans [5..30]
❷ T=[ ]
Pour i de 1 à N Faire
Répéter
Ecrire("T[",i,"]="), Lire(T[i])
Jusqu'à T[i] div 100 dans [1 .. 9]
Fin pour

Le Tableau de Déclaration des Nouveaux Types

Type
Bac = Tableau de 30 entiers

Le tableau de déclaration des objets globaux

Objet	Type / Nature	Rôle
I	Octet	
N	Octet	
T	Bac	
Circulaire	Fonction	

Analyse de la fonction Circulaire

Def Fn Circulaire (K :Entier) :Booléen

Résultat = Circulaire
❸ Circulaire ← Fn Premier(R[i]) = Vrai
❶ R= Proc Init(K, R)
❷ i = [i←0]
Répéter
i←i+1
Jusqu'à (Fn Premier(R[i])=Faux)ou(i=12)

Le Tableau de Déclaration des Nouveaux Types

Type
Tab = Tableau de 12 entiers

Le tableau de déclaration des objets locaux

Objet	Type / Nature	Rôle
I	Octet	
R	Tab	
Init	Procédure	
Premier	Fonction	

Analyse de la procédure Init

Def Proc Init (K :Entier ; Var R :Tab)

Résultat = R

❶ R = [c ← K div 100, d← K mod 100 div 10, u← K mod 10,
R[1]← K, R[2] ← c*100+u*10+d, R[3] ← d*100+c*10+u,
R[4] ← d*100+u*10+c, R[5] ← u*100+c*10+d, R[6] ← u*100+d*10+c]
Pour i de 7 à 12 Faire
R[i]← R[i-6] div 10
Fin Pour

Le tableau de déclaration des objets locaux

Objet	Type / Nature	Rôle
C	Octet	
D	Octet	
U	Octet	
i	Octet	

Analyse de la fonction Premier

Def Fn Premier (K :Entier) :Booléen

Résultat = Premier
❶ Premier ← test
❷ test = [ ]
Si K<2 Alors test ← Faux
Sinon si K=2 Alors test ← Vrai
Sinon si K mod 2 = 0 Alors test ← Faux
Sinon
i←1
Répéter
i←i+2
Jusqu'à (i>Racinecarré(K)) ou (K mod i =0)
Test ← i>Racinecarré(K)
Fin Si

Le tableau de déclaration des objets locaux

Objet	Type / Nature	Rôle
i	Octet	
Test	Booléen	

La traduction Pascal:

```
Program PremierCirculaire;
uses wincrt;
```

```
type Bac = array[1..30]of integer;
var t:Bac;
n,i:byte;
                    {******* la fonction premier *******}
function premier (k:integer):boolean;
var i:integer; test:boolean;
begin
if k=2 then test:=true
        else if (k mod 2=0)or(k<2) then test:=false
                else begin
                                i:=1;
                                repeat
                                i:=i+1;
                                until (k mod i=0)or(i>sqrt(k));
                                test := i>sqrt(k);
                        end;
premier:=test;
end;
                    {******* la fonction circulaire *******}
function circulaire(k:integer):boolean;
type Tab = Array[1..12] of integer;
var R:Tab ;
i:byte;
                    {******* la procedure Init *******}
                Procedure Init(K:integer; var R:Tab);
                var i,c,d,u:byte;
                Begin
                c:=k div 100;
                d:=k mod 100 div 10;
                u:=k mod 10;
                R[1]:=k;
                R[2]:=c*100+u*10+d;
                R[3]:=d*100+c*10+u;
                R[4]:=d*100+u*10+c;
                R[5]:=u*100+c*10+d;
                R[6]:=u*100+d*10+c;
                for i:= 7 to 16 do R[i]:=R[i-6]div 10;
                end;
begin
Init(K, R);
i:=0;
repeat
i:=i+1;
until (premier(R[i])=false)or (i=12);
circulaire:=premier(R[i])=true;
end;
                    {******* le programme principal *******}
begin
        repeat
                write('N = '); readln(N);
        until N in [5..30];
```

```
for i:=1 to N do
        repeat
                write('T[',i,'] = ');readln(T[i]);
        until (T[i] div 100 in [1..9]);
for i:=1 to n do
        if Circulaire(t[i]) then write(T[i]:5);
end.
```

Séance2-Sujet3

Un nombre est dit riche si au moins un de ses facteurs premiers est répété deux fois ou plus dans la décomposition du nombre en facteurs premiers.

Exemples :

- Le nombre **72** est dit **riche**, car 2 et 3 se répètent plus qu'une fois dans sa décomposition en facteurs premiers ($72=2^3*3^2$).
- Le nombre **22 n'est pas riche**, car tous ses facteurs premiers existent une seule fois (22 = 2*11).

Travail demandé :

Ecrire un programme Pascal qui permet de remplir un tableau **T** par **N** (**3 < N < 10**) entiers positifs non nuls à deux chiffres ou à trois chiffres, de trouver et d'afficher le ou les nombre(s) riche(s) du tableau **T**.

Exemple :

Pour **N = 6** et le tableau **T** suivant :

T	22	15	90	540	30	72
	1	2	3	4	5	6

Le programme affiche : "les nombres riches sont : 90, 540, 72"

En effet, 22=2*11, 15 = 3*5, **90 - 2*3*3*5**, **540 = 2*2*3*3*3*5**, 30 = 2*3*5 et **72=2*2*2*3*3**

Analyse du programme principal

❸ ***Résultat*** = [ ]

```
        Pour i de 1 à n Faire
                Si Fn Riche(T[i])=Vrai Alors Ecrire(T[i])
                Fin Si
        Fin Pour
```

❶ N= [ ]

```
        Répéter
                Ecrire ("N = "), Lire(N)
        Jusqu'à N dans [4..9]
```

❷ T = [ ]

```
        Pour i de 1 à N Faire
                Répéter
                Ecrire("T[",i,"]="), Lire(T[i])
                Jusqu'à (T[i]>9) et (T[i]<1000)
```

Le Tableau de Déclaration des Nouveaux Types

Type
Tablo = Tableau de 9 entiers

Le tableau de déclaration des objets globaux

Objet	Type / Nature	Rôle
I	Octet	
N	Octet	
T	Tablo	
Riche	Fonction	

Analyse de la fonction Riche

Def Fn Riche (K :Entier) :Booléen

Résultat = Riche
Riche ← Test
❷ Test ← (Fn Premier(d)=Vrai) et (K mod carré(d)=0)
❶ d = [d←1]
Répéter
d←d+1
Jusqu'à (Fn Premier(d)=Vrai) et (K mod carré(d)=0) ou (d>Racinecarré(k))

Le tableau de déclaration des objets locaux

Objet	Type / Nature	Rôle
D	Octet	
Test	Boolén	
Premier	Fonction	

Analyse de la fonction Premier

Def Fn Premier (K :Entier) :Booléen

Résultat = Premier
❶ Premier ← test
❷ test = [ ]
Si K<2 Alors test ← Faux
Sinon si K=2 Alors test ← Vrai
Sinon si K mod 2 = 0 Alors test ← Faux
Sinon
i←1
Répéter
i←i+2
Jusqu'à (i>Racinecarré(K)) ou (K mod i =0)
Test ← i>Racinecarré(K)
Fin Si

Le tableau de déclaration des objets locaux

Objet	Type / Nature	Rôle
I	Octet	
Test	Booléen	

La traduction Pascal:

```
program NombreRiches;
uses wincrt;
type tablo=array[1..9]of integer;
var t:tablo;
i,n:byte;
                    {******* la fonction premier *******}
```

```
function premier (k:integer):boolean;
var i:integer; test:boolean;
begin
if k=2 then test:=true
        else if (k mod 2=0)or(k<2) then test:=false
                else begin
                        i:=1;
                        repeat
                        i:=i+1;
                        until (k mod i=0)or(i>sqrt(k));
                        test := i>sqrt(k);
                    end;
premier:= test;
end;
                                {******* la fonction riche *******}
function Riche(K:integer):boolean;
var d:integer; test:boolean;
begin
d:=1;
repeat
        d:=d+1;
until (Premier(d))and(K mod sqr(d) =0) or (d>sqrt(k));
test:= (Premier(d))and(K mod sqr(d) =0);
Riche:=test;
end;
                                {******* le programme principal *******}
begin
repeat
write('N = '); readln(n);
until n in [4..9];
for i:=1 to n do
repeat
write('T[',i,'] = '); readln (t[i]);
until (t[i] >9) or (t[i]<1000);
for i := 1 to n do
if Riche(t[i]) then write(t[i]:5);
end.
```

Séance2-Sujet4

Un nombre **X** est dit **porte-bonheur** si ses chiffres forment une suite arithmétique de raison **r**, c'est-à-dire les chiffres de **X** sont ordonnés et la différence entre deux chiffres successifs est égale à une constante **r**.

Exemples :

- **1357** est un nombre **porte-bonheur** car ses chiffres sont ordonnés dans l'ordre croissant :
  1 < 3 < 5 < 7 et la différence entre deux chiffres successifs est constante et égale à **2**.

- **8765** est un nombre **porte-bonheur** car ses chiffres sont ordonnés dans l'ordre décroissant : **8 > 7 > 6 > 5** et la différence entre deux chiffres successifs est constante et égale à **-1**.
- **3679 n'est pas** un nombre **porte-bonheur** car la différence entre deux chiffres successifs n'est pas égale à une constante.

Travail demandé :

Ecrire un programme en Pascal qui permet de remplir un tableau **T** de **N (5≤ N ≤30)** entiers positifs de quatre chiffres, de chercher et d'afficher tous les entiers **porte-bonheur** du tableau **T**.

Analyse du programme principal

❸ ***Résultat*** **=** Proc Affichage (T,N)
❷ T = [ ]
Pour i de 1 à N Faire
Répéter
Ecrire("T[",i,"]="), Lire(T[i])
Jusqu'à T[i] div 1000 dans [1 .. 9]
Fin Pour
❶ N= [ ]
Répéter
Ecrire("N = "), Lire(N)
Jusqu'à N dans [5..30]

Le Tableau de Déclaration des Nouveaux Types

Type
Bac = Tableau de 30 entiers

Le tableau de déclaration des objets globaux

Objet	Type / Nature	Rôle
I	Octet	
N	Octet	
T	Bac	
Affichage	Procédure	

Analyse de la procédure Affichage

Def Proc Affichage (R: Bac; K :Entier;)

Résultat= [ ]
Pour i de 1 à K Faire
Si Fn Bonheur(R[i]) Alors Ecrire(R[i])
Fin Si
Fin Pour

Le tableau de déclaration des objets locaux

Objet	Type / Nature	Rôle
I	Octet	
Bonheur	Fonction	

Analyse de la fonction Bonheur

Def Fn Bonheur (X :Entier): booléen

Résultat= Bonheur
❸ Bonheur🡐 test
❷ Test 🡐 diff = ord(Ch[2])-ord(Ch[1])
❶ diff = [convch(X, Ch), i🡐1]
Répéter
I🡐 i+1
Diff🡐 ord(Ch[i+1])-ord(Ch[i])
Jusqu'à (Diff <> ord(Ch[2])-ord(Ch[1])) ou (i =long(ch)-1)

Le tableau de déclaration des objets locaux

Objet	Type / Nature	Rôle
I	Octet	
Diff	Octet	
Ch	Chaine	
Test	Booléen	

La traduction Pascal:

```
Program PorteBonheur;
uses wincrt;
Type Bac=Array[1..30]of integer;
var T:Bac; i,n:byte;
                    {******* la fonction bonheur *******}
function bonheur(x:integer):boolean;
var test:boolean;   i,diff:byte;    ch:string;
begin
str(x,ch);
i:=1;
repeat
i:=i+1;
diff:=ord(ch[i+1])-ord(ch[i]);
until (diff<>ord(ch[2])-ord(ch[1]))or(i=length(ch)-1);
test:=diff=ord(ch[2])-ord(ch[1]);
bonheur:=test;
end;
                    {******* la procédure  affichage *******}
procedure affichage(R:Bac;K:byte);
var i:byte;
begin
for i:=1 to k do
   if bonheur(R[i]) then writeln(R[i]:5);
end;
                    {******* le programme principal *******}
begin
repeat
write('N = ');Readln(n);
until n in [5..30];
for i:=1 to n do
```

```
repeat
write('T[',i,']=');readln(t[i]);
until t[i] div 1000 in [1..9];
Affichage(t,n);
end.
```

Séance3-Sujet1

Un mot **Zig-Zag** est un mot composé seulement par des lettres majuscules et dont l'ordre alphabétique de ses lettres croissent et décroissent ou inversement d'une façon alternative.

Exemples :

- Le mot **ADAM** est dit **Zig-Zag**, car l'ordre alphabétique de "**A**" est inférieur à celui de "**D**" dont son ordre alphabétique est supérieur à celui de "**A**" qui le suit et l'ordre alphabétique de "A" est inférieur à celui de "**M**".
- Le mot "**RANIM**" est dit **Zig-Zag**, car l'ordre alphabétique de "**R**" est supérieur à celui de "**A**" dont son ordre alphabétique est inférieur à celui de "**HN**" et l'ordre alphabétique de "**N**"' est supérieur à celui de "**I**" dont son ordre alphabétique est inférieur à celui de "**M**.
- Le mot "**PROGRAMME**" est dit **non Zig-Zag**, car l'ordre alphabétique de "**B**" est inférieur à celui de "**A**" dont son ordre alphabétique est supérieur à celui de "**O**" dont son ordre alphabétique est supérieur à celui de "**G**".
- Le mot "**BACCALAUREAT**" est dit **non Zig-Zag**, car l'ordre alphabétique de "**B**" est supérieur à celui de "**A**" dont son ordre alphabétique est inférieur à celui de "**C**" et l'ordre alphabétique de "**C**" est égal à celui du caractère qui le suit ("**C**").

Travail demandé :

Ecrire un programme Pascal qui permet de remplir un tableau **T** par **N** (**5≤ N ≤10**) mots composés par des lettres majuscules et dont leurs longueurs sont comprises entre **2** et **12** et d'afficher les mots **ZigZag** du tableau **T**.

Analyse du programme principal

❸ ***Résultat*** = [ ]
```
    Pour i de 1 à N Faire
        Si Fn Zigzag(T[i]) Alors Ecrire(T[i])
        Fin Si
    Fin Pour
```
❶ N = [ ]
```
    Répéter
        Ecrire("N = "), Lire(N)
    Jusqu'à N dans [5..10]
```
❷T = [ ]
```
    Pour i de 1 à N Faire
        Répéter
            Ecrire("T[ ",i," ]= "), Lire(T[i])
        Jusqu'à (Long(T[i]) dans [2..12]) et (Fn Verif(T[i])=Vrai)
```
Le Tableau de Déclaration des Nouveaux Types

Type
Bac = Tableau de 30 Chaînes [12]

Le tableau de déclaration des objets globaux

Objet	Type / Nature	Rôle
I	Octet	
N	Octet	
T	Bac	
ZigZag	Fonction	
Verif	Fonction	

Analyse de la fonction Verif

Def Fn Verif(X :Chaîne) : Booléen

Résultat = Verif
❸ Verif ←Test
❷ Test ← X[i] dans ["A".."Z"]
❶ I = [i←0]
Répéter
I←i+1
Jusqu'à (Non(X[i] dans ["A".."Z"])) ou (i=Long(X))

Le tableau de déclaration des objets locaux

Objet	Type / Nature	Rôle
I	Octet	Compteur
Test	Booléen	Aura comme valeur **Faux** si le dernier caractère testé n'est pas une lettre majuscule ou **Vrai** dans le cas contraire

Analyse de la fonction ZigZag

Def Fn ZigZag(X :Chaîne) : Booléen

Résultat = ZigZag
❸ ZigZag ←Test
❷ Test ← ((X[i]>X[i-1]) et(X[i]>X[i+1])) ou ((X[i]<X[i-1]) et(X[i]<X[i+1]))
❶ I = [i←1]
Répéter
I←i+1
Jusqu'à ((X[i]>X[i-1]) et(X[i]<X[i+1])) ou ((X[i]<X[i-1]) et(X[i]>X[i+1]))
ou (i=Long(X)-1)

Le tableau de déclaration des objets locaux

Objet	Type / Nature	Rôle
I	Octet	Compteur
Test	Booléen	

La traduction Pascal:

```
Program NomZigZag;
uses wincrt;
type Bac = array[1..10] of string[12];
var n,i:Byte;  t:Bac;
                    {******* la fonction Verif *******}
function verif(x:string):boolean;
var i:Byte;
```

```
test:boolean;
begin
i:=0;
repeat
i:=i+1;
until (not(x[i] in ['A'..'Z']))or (i=length(x));
test:=x[i] in ['A'..'Z'];
verif:=test;
end;
```

{**** la fonction ZigZag *******}***

```
function ZigZag(x:string):boolean;
var i:integer;
test:boolean;
begin
i:=1;
repeat
i:=i+1;
until ((x[i]>x[i-1])and(x[i]<x[i+1])) or ((x[i]<x[i-1])and(x[i]>x[i+1])) or (i = length(x)-1);
test:= ((x[i]>x[i-1])and(x[i]>x[i+1])) or ((x[i]<x[i-1])and(x[i]<x[i+1]));
zigzag :=test;
end;
```

{**** le programme principal *******}***

```
begin
repeat
write('N = '); readln(n);
until n in [5..10];
for i:= 1 to n do
repeat
write('t[',i,'] = ');
readln(t[i]);
until (length(t[i]) in [2..12]) and (verif(t[i]));
for i:=1 to n do
if ZigZag(t[i]) then writeln(t[i]);
end.
```

Séance3-Sujet2

On définit le **D**egré de **R**essemblance **DR** entre deux mots de même longueur par la formule suivante :

DR = (nombre de caractères en communs bien placés / longueur du mot) * 100

NB : Un caractère est dit bien placé lorsqu'il occupe la même position dans les deux mots.

Exemples :

- Pour mot1= "**EX**E**M**PLE" et mot2 = "**EX**A**M**ENS"
  Le degré de ressemblance DR = (3 / 7)* 100 = 42.85
- Pour mot1 = "TRAITEMENTS" et mot2 - "INFORMATION"
  Le degré de ressemblance DR = (0/1 1* 100 = 00.00

Travail demandé :

Ecrire un programme Pascal qui permet de saisir une chaîne **Ch** non vide et composée de lettres majuscules, puis de remplir un tableau **T** par **N (5 ≤N ≤10)** chaînes de

caractères composées de lettres majuscules et de même longueur que **Ch** et d'afficher le degré de ressemblance entre **Ch** et les éléments de **T**.

Analyse du programme principal

❹ ***Résultat*** = [ ]
    Pour i de 1 à N Faire
        Ecrire(Ch," ",T[i]," ", Fn Degre(Ch,T[i]))
    Fin Pour
❶ N = [ ]
    Répéter
    Ecrire("N = "), Lire(N)
    Jusqu'à N dans [5..10]
❷ Ch = [ ]
    Répéter
    Ecrire("Ch = "), Lire(Ch)
    Jusqu'à (Fn Majuscule(Ch)=Vrai) et (Long(Ch)>0)
❸ T = [ ]
    Pour i de 1 à N Faire
        Répéter
            Ecrire("T[",i,"]= "), Lire(T[i])
        Jusqu'à (Fn Majuscule(T[i])) et (Long(T[i])=Long(Ch))
    Fin Pour

Le Tableau de Déclaration des Nouveaux Types

Type
Bac = Tableau de 10 Chaînes

Le tableau de déclaration des objets globaux

Objet	Type / Nature	Rôle
Ch	Chaîne	
N	Octet	
i	Octet	
T	Bac	
Majuscule	Fonction	
Degre	Fonction	

Analyse de la fonction Majuscule

Def Fn Majuscule(Ph:Chaîne) : Booléen

Résultat = Majuscule
❸ Majuscule ←Test
❷ Test ← Ph[i] dans ["A".."Z"]
❶ I = [i←0]
    Répéter
    I←i+1
    Jusqu'à (Non(Ph[i] dans ["A".."Z"])) ou (i=Long(Ph))

Le tableau de déclaration des objets locaux

Objet	Type / Nature	Rôle
I	Octet	Compteur
Test	Booléen	

Analyse de la fonction Degre

Def Fn Degre(Ch,X :Chaîne) : Réel

Résultat = Degre
❸ Degre←D
❷ D←nb*100/Long(x)
❶ Nb = [nb←0]
Pour i de 1 à Long(x) Faire
Si X[i] = Ch[i] Alors nb←nb+1
Fin Si
Fin Pour

Le tableau de déclaration des objets locaux

Objet	Type / Nature	Rôle
I	Octet	Compteur
Nb	Octet	
D	Réel	

La traduction Pascal:

```
program Ressemblance;
uses wincrt;
type Bac=array[1..10]of string;
var ch:string;
t:Bac; n,i:byte;
                    {******* la fonction Majuscule *******}
Function Majuscule(Ph:string):boolean;
var i:byte ; test:boolean;
begin
i:=0;
repeat
i:=i+1;
until (not(Ph[i]in['A'..'Z']))or (i=length(Ph));
test:=Ph[i]in['A'..'Z'];
Majuscule:=test;
end;
                    {******* la fonction Degre *******}
Function degre(Ch,X:string):real;
var i,nb:byte ; d:real;
begin
nb:=0;
for i:= 1 to length(x) do
if x[i]=ch[i] then nb:=nb+1;
d:=nb*100/length(x);
degre:=d;
end;
                    {******* le programme principal *******}
begin
repeat
write('N = ');readln(n);
until n in [5..10];
repeat
```

```
write('Ch = '); readln(ch);
until (Majuscule(ch)=true) and (Length(ch)>0);
for i:= 1 to n do
repeat
write('T[',i,'] = ');readln(t[i]);
until (Majuscule(T[i]))and(length(T[i])=length(ch));
for i:=1 to n do
writeln(ch,' ',t[i],' ', Degre(t[i],ch):5:2);
end.
```

Séance3-Sujet3

Une chaîne est dite existante dans un tableau de chaînes si elle peut être formée à partir de la concaténation des i^{èmes} caractères des différents éléments de ce tableau.

Exemple :

Pour **N = 5** et le tableau **T** suivant :

T	"SALAH"	"AMIRA"	"BILEL't	"ANWAR$^{t'}$	"KARIM"
	1	**2**	**3**	**4**	**5**

- Pour **Ch** = "AMINA" le programme affiche : "**chaine existante dans T**" car elle est le ***Résultat*** de la concaténation des 2**èmes** caractères des différents éléments de **T**.
- Pour **Ch** = "SALWA" le programme affiche : "**chaine inexistante dans T**" car les caractères de **Ch** n'existe pas dans la même position dans les éléments de **T**.
- Pour **Ch** = "HAMZA" le programme affiche : "**chaine inexistante dans T**"car aucune concaténation des i^{èmes} caractères de **T** ne forme la chaine **Ch**.

Travail demandé :

Ecrire un programme Pascal qui permet de saisir un entier **N** (**5≤ N ≤40**) et une chaîne **Ch** composée de lettres majuscules et de longueur **N**, puis de remplir un tableau **T** par **N** chaînes composées de lettres majuscules et de même longueur que **Ch** et de vérifier l'existence de **Ch** dans **T** comme décrit ci-dessus.

Analyse du programme principal

```
❹ Résultat = [ ]
        Si Fn Existe(Ch,T,N) = Vrai Alors Ecrire(Ch," Existe dans le tableau")
                                    Sinon Ecrire(Ch, "n'existe pas dans le tableau")
        Fin Si
❸Ch = [ ]
        Répéter
        Ecrire("Ch = "), Lire(Ch)
        Jusqu'à (Fn Verif(Ch)) et (Long(Ch)=N)
❷T = [ ]
        Pour i de 1 à N Faire
                Répéter
                        Ecrire("T[ ",i," ]= "), Lire(T[i])
                Jusqu'à (Fn Verif(T[i])) et (Long(T[i])=n)
        Fin Pour
```

❶ N = [ ]
Répéter
Ecrire("N = "), Lire(N)
Jusqu'à N dans[5..10]

Le Tableau de Déclaration des Nouveaux Types

Type
Bac = Tableau de 10 Chaînes[10]

Le tableau de déclaration des objets globaux

Objet	Type / Nature	Rôle
Ch	Chaîne	
N	Octet	
T	Bac	
i	Octet	
Verif	Fonction	
Existe	Fonction	

Analyse de la fonction Verif

Def Fn Verif(Ph:Chaîne) : Booléen

Résultat = Verif
❸ Verif ←Test
❷ Test ← Ph[i] dans ["A".."Z"]
❶ I [i←0]
Répéter
I←i+1
Jusqu'à (Non(Ph[i] dans ["A".."Z"])) ou (i=Long(Ph))

Le tableau de déclaration des objets locaux

Objet	Type / Nature	Rôle
I	Octet	Compteur
Test	Booléen	

Analyse de la fonction Existe

Def Fn Existe(Ch:Chaîne; T:Bac;N:Entier) : Booléen

Résultat = Existe;
❸ Existe←Test
❷ Test ←X=Ch
❶ X = [i←0]
Répéter
X←""
I←i+1
Pour J de 1 à N Faire
X←X+T[j][i]
Fin Pour
Jusqu'à (X=Ch) ou (i=N)

Le tableau de déclaration des objets locaux

Objet	Type / Nature	Rôle
I	Octet	Compteur
J	Octet	

Test	Booléen	
X	Chaine [10]	

La traduction Pascal:

```
program Existence;
uses wincrt;
type bac=array[1..10] of string[10];
var ch:string[10];  t:bac;  i,n:integer;
                        {******* la fonction Verif *******}
function verif (Ph:string):boolean;
var test:boolean; i:byte;
begin
i:=0;
repeat
i:=i+1;
until not(Ph[i] in ['A'..'Z']) or (i=length(Ph));
Test:=Ph[i] in ['A'..'Z'];
Verif:=test;
end;
                        {******* la fonction Existe *******}
function existe(ch:string;t:bac; n:integer):boolean;
var i,j:byte; x:string[10]; test:boolean;
begin
i:=0;
repeat
x:='';
i:=i+1;
for j:=1 to n do
x:=x+t[j][i];
until (x=ch) or (i=n);
test:= x=ch;
existe:=test;
end;
                        {******* le programme principal *******}
begin
repeat
write('N = '); readln(n);
until n in [5..10];
for i:=1 to n do
repeat
write('T[ ',i,' ] = '); readln(t[i]);
until (verif(t[i]) )and (length(t[i])=n);
repeat
write('Ch = '); readln(ch);
until verif(ch) and (length(ch)=n);
if existe(Ch, T,N) then writeln(Ch,' Existe dans le tableau ')
else writeln(Ch, ' n''existe pas dans le tableau');
end.
```

Séance3-Sujet4

Un entier est dit **Zig-Zag** lorsque ses chiffres croissent et décroissent ou inversement d'une façon alternative.
Exemples :

- Le nombre **13254** est dit **Zig-Zag**, car **1** est inférieur à **3** qui est supérieur à **2** et **2** est inférieur à 5 qui est supérieur à **4**.
- Le nombre **8781** est dit **Zig-Zag**, car **8** est supérieur à **7** et **7** est inférieur à **8** qui est supérieur à **1**.
- Le nombre **8761** est dit **non Zig-Zag**, car **8** est supérieur à **7** qui est aussi supérieur à **6**.
- Le nombre **2997** est dit **non Zig-Zag** car **2** est inférieur à **9** qui est égal au chiffre qui le suit (**9**).

Travail demandé :

Ecrire un programme Pascal qui permet de remplir un tableau **T** par **N** (**5 ≤N≤ 25**) entiers positifs composés d'au minimum trois chiffres et d'afficher les nombres **Zig-Zag** du tableau **T**.

Analyse du programme principal

❸ ***Résultat*** = [ ]
Pour i de 1 à N Faire
Si Fn ZigZag(T[i]) Alors Ecrire(T[i])
Fin Si
❶N = [ ]
Répéter
Ecrire("N = "), Lire(N)
Jusqu'à N dans [5..25]
❷ T = [ ]
Pour i de 1 à N Faire
Répéter
Ecrire("T[",i,"] ="), Lire(T[i])
Jusqu'à T[i]>99
Fin Pour

Le Tableau de Déclaration des Nouveaux Types

Type
Tp = Tableau de 25 entiers

Le tableau de déclaration des objets globaux

Objet	Type / Nature	Rôle
N	Octet	
T	Tp	
I	Octet	
ZigZag	Fonction	

Analyse de la fonction ZigZag

Def Fn ZigZag(Z:Entier) : Booléen

Résultat = ZigZag
❸ ZigZag ←Test

❷ Test ← ((X[i]>X[i-1]) et(X[i]>X[i+1])) ou ((X[i]<X[i-1]) et(X[i]<X[i+1]))
❶ I = [i←1, Convch(Z,X)]
Répéter
I←i+1
Jusqu'à ((X[i]>X[i-1]) et(X[i]<X[i+1])) ou ((X[i]<X[i-1]) et(X[i]>X[i+1]))
ou (i=Long(X)-1)

Le tableau de déclaration des objets locaux

Objet	Type / Nature	Rôle
I	Octet	Compteur
X	Chaîne	
Test	Booléen	

La traduction Pascal:

```
Program EntierZigzag;
uses wincrt;
type tab = array[1..25] of integer;
var n,i:byte; t:tab;
                    {******* la fonction ZigZag *******}
function Zigzag(Z:integer):boolean;
var i:byte;X:string; test:boolean;
begin
str(Z,X);i:=1;
repeat
i:=i+1;
until ((x[i]>x[i-1])and(x[i]<x[i+1])) or ((x[i]<x[i-1])and(x[i]>x[i+1])) or
 (i = length(x)-1);
test := ((x[i]>x[i-1])and(x[i]>x[i+1])) or ((x[i]<x[i-1])and(x[i]<x[i+1]));
zigzag:=test;
end;
                    {******* le programme principal *******}
begin
repeat
write('N = '); readln(n);
until n in [5..25];
for i:= 1 to n do
repeat
write('T[ ',i,' ] = '); readln(t[i]);
until (T[i]) >99;
for i:=1 to n do
begin
if Zigzag(T[i]) then writeln(t[i]);
end;
end.
```

Séance4-Sujet1

Un nombre **P** est appelé **k-parfait** si et seulement si la somme de tous les diviseurs positifs de **P**, y compris 1 et lui-même, est égale à **k * P**. Avec **k** un entier naturel donnée.

Exemple :

- Le nombre 28 est **2-parfait**, car la somme de ses diviseurs est 56 = 2*28.

- Le nombre 120 est **3-parfait**, car la somme de ses diviseurs est 360 3*120.

Travail demandé :
Ecrire un programme Pascal qui permet de chercher et d'afficher tous les nombres de l'intervalle **[N, M]** avec **10 < N ≤ M < 31000** qui sont **2-parfaits** suivis par ceux qui sont **3-parfaits** sur une autre ligne et ceux qui sont **4-parfaits** sur une autre ligne.

Analyse du programme principal

❷ ***Résultat*** = Proc Afficher(N,M)
❶ (N,M) =[ ]
Répéter
Ecrire("N = "), lire(N)
Ecrire("M = "), lire(M)
Jusqu'à (N>10) et (N<=M) et (M<31000)

Le tableau de déclaration des objets globaux

Objet	Type / Nature	Rôle
N	Entier	
M	Entier	
Affiche	Procédure	

Analyse de la procédure Afficher

Def Proc Afficher (Inf,Sup :Entier)

❶ ***Résultat*** = [ ]
Pour k de 2 à 4 Faire
Pour i de Inf à Sup Faire
Si Fn SomDiv(i)=k*i Alors Ecrire(i,"␣")
Fin Si
Fin Pour
Fin Pour

Le tableau de déclaration des objets locaux

Objet	Type / Nature	Rôle
I	Entier	
K	Octet	
SomDiv	Fonction	

Analyse de la fonction SomDiv

Def Fn SomDiv (K:Entier) :EntierLong

Résultat = SomDiv
❷ SomDiv←S
❶ S= [S← K+1]
Pour i de 2 à K div 2 Faire
Si K mod i = 0 Alors S←S+i
Fin Si
Fin pour

Le tableau de déclaration des objets locaux

Objet	Type / Nature	Rôle
I	Entier	
S	Entier Long	

La traduction Pascal:

```
program Kparfait;
uses wincrt;
var n,m:integer;
                    {******* la fonction somdiv *******}
function somdiv(k:integer):longint;
var     s:longint; i:integer;
begin
s:=k+1;
for i:=2 to k div 2  do
if k mod i =0 then s:=s+i;
somdiv:=s;
end;
                    {******* la procédure afficher *******}
procedure afficher(inf, sup:integer);
var i :integer;k:byte;
begin
for k:=2 to 4 do
begin
for i:=inf  to sup do
if somdiv(i)=i*k then write(i,' ');
writeln;
end;
end;
                    {******* le programme principal *******}
begin
repeat
write('N = ');readln(n);
write('M = ');readln(m);
until (10<n) and (n<=m) and (m<31000);
afficher(n,m);
end.
```

Séance4-Sujet2

Soient **N** et **M** deux entiers naturels, on dit que **N** et **M** sont **homogènes** s'ils admettent les mêmes facteurs premiers.

Exemples :

- **N = 60** et **M = 90 sont homogènes**, car ils ont les mêmes facteurs premiers qui sont 2, 3 et 5.
  En effet, $60 = 2^2 * 3 * 5$ et $90 = 2 * 3^2 * 5$
- **N = 60** et **M = 420 ne sont pas homogènes**, car ils n'ont pas les mêmes facteurs premiers.
  En effet, $60 = 2^2 * 3 * 5$ et $420 = 2^2 * 3 * 5 * 7$

N.B. :

On dit qu'un nombre **a** admet le nombre **b** comme facteur premier lorsque **b** est un nombre premier qui divise **a**.

Travail demandé :

Ecrire un programme Pascal qui permet de saisir deux entiers N et M (**5 ≤ N < M**), de vérifier et d'afficher s'ils sont homogènes ou non.

Analyse du programme principal

❷ ***Résultat*** = [ ]

Si Fn Homgene(N,M) Alors Ecrire(N," et ",M," sont Homogènes")
Sinon Ecrire(N," et ",M," ne sont pas Homogènes")
Fin Si

❶ (N, M) = [ ]

Répéter
Ecrire("N = "), Lire(N)
Ecrire("M = "), Lire(M)
Jusqu'à (N>=5) et(N<M)

Le tableau de déclaration des objets globaux

Objet	Type / Nature	Rôle
N	Entier	
M	Entier	
Homogene	Fonction	

Analyse de la fonction Homogene

Def Fn Homogene (N, M:Entier) :Booléen

Résultat = Homogene
❷ Homogene ←Test

❶ Test = [Test← Vrai,

Max←(N+M+Abs(N-M)) div 2]
Si Fn PGCD(N,M) =1 Alors Test ← Faux
Sinon D←1
Répéter
D←D+1
Si (Fn Premier(d)) Alors
Si (N mod d=0) ouex (M mod d=0) Alors Test← Faux
Fin Si
Fin Si
Jusqu'à (test=Faux) ou (d=Max div 2)
Fin Si

Le tableau de déclaration des objets locaux

Objet	Type / Nature	Rôle
D	Entier	
Max	Entier	
Test	Booléen	
PGCD	Fonction	
Premier	Fonction	

Analyse de la fonction Premier

Def Fn Premier (K :Entier) :Booléen

Résultat = Premier

❷Premier ← test
❶ test = [ ]
Si K<2 Alors test ← Faux
Sinon si K=2 Alors test ← Vrai
Sinon si K mod 2 = 0 Alors test ← Faux
Sinon
i←1
Répéter
i←i+2
Jusqu'à (i>Racinecarré(K)) ou (K mod i =0)
Test ← i>Racinecarré(K)
Fin Si

Le tableau de déclaration des objets locaux

Objet	Type / Nature	Rôle
I	Octet	
Test	booléen	

Analyse de la fonction PGCD

Def Fn PGCD (a,b :Entier) :Entier

Résulat = PGCD
❷ PGCD ← X
❶ X =[X←a, Y←b]
Tant que X <> Y Faire
Si X>Y Alors X ←X -Y
Sinon Y ← Y-X
Fin Si
Fin Tant que

Le tableau de déclaration des objets locaux

Objet	Type / Nature	Rôle
X	Entier	
Y	Entier	

La traduction Pascal:

```
program EntiersHomogenes;
uses wincrt;
var n,m:integer;
                    {******* la fonction pgcd *******}
function pgcd(a,b:integer):integer;
var x, y:integer;
begin
x:=a;  y:=b;
while (x<>y) do
if   x>y      then x:=x-y
     else y:=y-x;
pgcd:=x;
end;
                    {******* la fonction pemier *******}
function premier(k:integer):boolean;
```

```
var i:byte; test:boolean;
begin
if k<2 then test:= false
else if  k=2 then test:=true
else if k mod 2 =0 then test := false
else
        begin
        i:=1;
        repeat
        i:=i+2;
        until (k mod i =0) or (i>sqrt(k));
        test:= i>sqrt(k);
        end;
premier:=test;
end;
```

{**** la fonction homogene *******}***

```
function homogene(n,m:integer):boolean;
var d, max :integer; test :boolean;
begin
max:=(n+m+abs(n-m))div 2;
test:=true;
if pgcd(n,m)=1 then test:=false
else
begin
d:=1;
repeat
d:=d+1;
if premier(d) then
  if (n mod d=0) xor (m mod d=0)then test := false;
until (test=false) or (d=max div 2);
end;
homogene:=test;
end;
```

{**** le programme principal *******}***

```
begin
repeat
write('N = ');readln(n);
write('M = ');readln(m);
until (N>=5) and(N<M);
if homogene(n,m) then writeln(n,' et  ',m,' sont homogènes')
else writeln(n,' et  ',m,' ne sont pas homogènes');
end.
```

Séance4-Sujet3

La décomposition du **PGCD(A,B)** en facteurs premiers (avec **A ≥ 2** et **B ≥ 2**) est le produit des facteurs premiers apparaissant à la fois dans la décomposition de **A** et de **B** munis du **plus petit** des exposants trouvés dans la décomposition de **A** et de **B**.
NB : On dit qu'un nombre **a** admet le nombre **b** comme facteur premier lorsque **b** est un nombre premier qui divise **a**.

Travail demandé :

Ecrire un programme Pascal qui permet de saisir deux entiers **A** et **B** (**10≤ A ≤B ≤10000**) de chercher et d'afficher la décomposition en facteurs premiers du **PGCD(A,B)** en utilisant le principe décrit ci-dessus.

Exemple :

Pour **A** = 378 et **B** = 8820

Liste des facteurs premiers de **A** = 378 = $2 * 3^3 * 7$

Liste des facteurs premiers de **B** = 8820 = $2^2 * 3^2 * 5 * 7^2$

Alors le programme affiche : PGCD (378, 8820) = $2 * 3^2 * 7 = 126$

Analyse du programme principal

❷ ***Résultat*** = Ecrire ("Le PGCD(",a," ," ,b,") = ",Fn PGCD (a,b))

❶ (a,b) = [ ]

Répéter

Ecrire("A = "), Lire(a)

Ecrire("B = "), Lire(b)

Jusqu'à (10<=a) et (a<=b) et(b<=10000)

Le tableau de déclaration des objets globaux

Objet	Type / Nature	Rôle
A	Entier	
B	Entier	
PGCD	Fonction	

Analyse de la fonction PGCD

Def Fn PGCD(A,B :Entier) :Entier

Résulat = PGCD

❷ PGCD ← R

❶ R =[R←1, d←2]

Répéter

Si (N mod d=0) et (M mod d = 0) Alors R←R*d, N←N div d, M← M div d

Sinon

Si N mod d =0 Alors N←N div d

Sinon Si M mod d = 0 Alors M ←M div d

Fin Si

D←d+1

Fin si

Jusqu'à (N=1) ou (M=1)

Le tableau de déclaration des objets locaux

Objet	Type / Nature	Rôle
R	Entier	
D	Entier	

La traduction Pascal:

```
Program PGCDde2Entiers;
Var A,B:Integer;
                    {******* la fonction pgcd *******}
Function PGCD(A,B:integer):integer;
var d,R:integer;
```

```
begin
d:=2; R:=1;
repeat
if (A mod d=0) and(B mod d=0)then begin R:=R*d; A:= A div d; B:= B div d; end
        else begin
                    if A mod d = 0 then A:= A div d
                            else if B mod d = 0 then B := B div d;
                    d:=d+1;
             end;
until (A = 1) or (B = 1);
PGCD:=R;
end;
                                {******* la programme principal *******}
Begin
repeat
write('A = '); Readln(A);
write('B = '); Readln(B);
Until (10<=A) and (A<=B) and (B<=10000);
Writeln ('Le PGCD de ', A, ' et ',B,' = ',PGCD(A,B));
End.
```

(PPCM[1] au lieu du PGCD)

Analyse du programme principal

❷ ***Résultat*** = Ecrire ("Le PPCM(",a," ," ,b,") = ",Fn PPCM (a,b))
❶ (a,b) = [ ]
Répéter
Ecrire("A = "), Lire(a)
Ecrire("B = "), Lire(b)
Jusqu'à (10<=a) et (a<=b) et(b<=10000)

Le tableau de déclaration des objets globaux

Objet	Type / Nature	Rôle
A	Entier	
B	Entier	
PPCM	Fonction	

Analyse de la fonction PPCM

Def Fn PPCM(N,M :Entier) :Entier long

Résulat = PPCM
❷ PPCM ← R
❶ R =[R←1, d←2]
Répéter
Si (N mod d=0) et (M mod d = 0) Alors R←R*d, N←N div d, M← M div d
Sinon
Si N mod d =0 Alors R←R * d, N←N div d

[1] ***A=378 = $2*3^3*7$ et B= 8820=$2^2*3^2*5*7^2$ → PPCM = $2^2*3^3*5*7^2$ = 26460***

Sinon Si M mod d = 0 Alors R← R * d, M ←M div d
Sinon D←d+1

Fin si
Jusqu'à (N=1) et (M=1)

Le tableau de déclaration des objets locaux

Objet	Type / Nature	Rôle
R	Entier Long	
d	Entier	

La traduction Pascal:

```
Program PPCMde2Entiers;
Var A,B: Integer;
                    {******* la fonction ppcm *******}
function PPCM(N,M:integer):longint;
var d:integer;
   r:longint;
begin
d:=2; R:=1;
repeat
if (N mod d=0) and (M mod d=0)then begin R:=R*d; N:= N div d; M:= M div d; end
else    if N mod d = 0 then begin N:= N div d; r:=r*d; end
    else if M mod d = 0 then begin M := M div d;r:=r*d; end
    else     d:=d+1;
until (N = 1) and (M = 1);
PPCM:=r;
end;
                    {******* la programme principal *******}
Begin
repeat
write('A = '); Readln(A);
write('B = '); Readln(B);
Until (A>=10)and(A<=B) and(B<=10000);
write('Le PPCM( ',a,' , ',b,' ) = ',PPCM(A,B));
End.
```

Séance4-Sujet4

Un nombre est dit **k-pp** (**Presque Premiers**), s'il s'écrit sous la forme d'un produit de **k** nombres premiers non nécessairement distincts.

Exemples :

219 = 3*73 est un nombre **2-pp** car il est le produit de deux nombres premiers.
32 2*2*2*2*2 est un nombre **5-pp** car il est le produit de cinq nombres premiers.
17 = 17 est un nombre **1-pp** car il est premier.

Travail demandé :

Ecrire un programme Pascal qui permet de remplir un tableau **T** par **N** (**5≤ N ≤50**) entiers positifs de 3 chiffres, de chercher et d'afficher les **k-pp** nombres du tableau **T**. Sachant que **k** est un entier choisi aléatoirement de l'intervalle [2,5].

Exemple :

Pour **N=5, k = 3** et le tableau **T** suivant :

T	231	846	187	722	490

Les nombres 231 et 722 sont dits **3-pp** et seront affichés puisque :

231= 3 * 7 * 11

722=2*19*19

N.B : Un nombre est dit premier s'il n'est divisible que par **1** et par lui-même. Par définition, 1 n'est pas un nombre premier

Analyse du programme principal

❹ ***Résultat*** = Ecrire("K = ", K)
Pour i de 1 à N faire
Si Fn Nombre(T[i]) =K Alors Ecrire (T[i],"␣")
Fin Si
Fin Pour
❸ K ← Aléa(4)+2
❶ N = [ ] Répéter
Ecrire("N = "), Lire(N)
Jusqu'à N dans [5..50]
❷ T = [ ] Pour i de 1 à N Faire
Répéter
Ecrire("T[",i,"]= "), Lire(T[i])
Jusqu'à (T[i] div 100 dans[1..9])
Fin Pour

Le Tableau de Déclaration des Nouveaux Types

Type
BacTp = Tableau de 50 entiers

Le tableau de déclaration des objets globaux

Objet	Type / Nature	Rôle
N	Octet	
K	Octet	
I	Octet	
T	BacTp	
Nombre	Fonction	

Analyse de la fonction Nombre

Def Fn Nombre (X :Entier) :Entier

Résulat = Nombre
❷ Nombre ←nb
❶ nb = [nb←0, d←2]
Répéter
Si X mod d = 0 Alors X ← X div d , Nb←Nb+1
Sinon d← d+1
Fin Si
Jusqu'à (X = 1)

Le tableau de déclaration des objets locaux

Objet	Type / Nature	Rôle
nb	Entier	
d	Entier	

La traduction Pascal:

```
program PresquePremier;
uses wincrt;
type Bactp=array[1..50]of integer;
var t:Bactp;
var n, i, k: byte;
                    {******* la fonction Nombre *******}
function Nombre(X:integer):Integer;
var nb,d:integer;
begin
d:=2;
nb:=0;
repeat
if X mod d = 0 then begin nb:=nb+1; X:=X div d; end
else d:=d+1;
until X = 1;
Nombre:=nb;
end;
                    {******* la programme principal *******}
begin
repeat
write('N = ');readln(n);
until n in [5..50];
for i :=1 to n do
        repeat
        write('T[ ',i,' ] ='); readln(t[i]);
        until t[i] div 100 in [1..9];
randomize;   k:=random(4)+2;
writeln('K = ',K);
for i:=1 to n do
if Nombre(t[i])=k then writeln(t[i],' ');
end.
```

Séance5-Sujet1

Pour accéder à son compte sur un site, un utilisateur doit disposer d'un identifiant **id** et d'un mot de passe **pw**. L'identifiant **id** doit être unique et le mot de passe **pw** doit contenir au moins six caractères.

Travail demandé :

Ecrire un programme pascal qui permet de :

- ✓ Remplir deux tableaux **Tid** et **Tpw** contenant respectivement **N** identifiants distincts et **N** mots de passe (**2 ≤ N ≤ 10**) de façon à ce que **Tid[i]** correspond à **Tpw[i]**.
- ✓ Vérifier l'accès à un compte donné et ce comme décrit ci-dessous :
  - Saisir un identifiant **id** et un mot de passe **pw**,
  - Afficher le message "**id, bienvenu sur notre site**" si l'**id** existe dans le tableau **Tid** et le **pw** correspondant est correct.
  - Afficher le message "**Vérifiez votre identificateur et/ou votre mot passe**" si l'**id** n'existe pas dans le tableau **Tid** ou le **pw** est incorrect.

Exemple :

Pour **N = 4** et

Tid	Azerty	Tunisial	ali58	soltan 1
Tpw	AZer12	DF4567edc	ALI58ali	00aqwZygN

1er cas

Id = Tunisial
Pw = DF4567edc
Le programme affiche : "Tunisia1, bienvenu sur notre site"

2ème cas

Id = Tunisia
Pw = DF4567edc
Le programme affiche : "Vérifiez votre identificateur et/ou votre mot de passe"

3ème cas

Id = Tunisia1
Pw = DF4567
Le programme affiche : "Vérifiez votre identificateur et/ou votre mot de passe"

Analyse du programme principal

❼ ***Résultat*** = [ ]
Si (P>0) et (Tpw[P]=Pw) Alors Ecrire(id, ", Bienvenue sur notre site")
Sinon Ecrire("Vérifiez votre identificateur et/ou votre mot de passe ")
Fin Si
❻ P←Fn Existe(id,1,N,Tid)
❸ Tpw = [Répéter Ecrire("Tpw[1]="),Lire(Tpw[1]) jusqu'à long(Tpw[1])>5]
Pour i de 2 à N Faire
Répéter Ecrire("Tpw[",i,"]="),Lire(Tpw[i]) jusqu'à long(Tpw[i])>5
Fin Pour
❺ Pw = [ ]
Répéter Ecrire("Pw : "),Lire(Pw) jusqu'à long(Pw)>5
❹ Id = [ ]
Répéter Ecrire("Id : "),Lire(Id) jusqu'à long(Id)>0
❶ N = [ ]
Répéter
Ecrire("N = "),Lire(N)
Jusqu'à N dans [2..10]
❷ Tid = [Répéter Ecrire("Tid[1]="),Lire(Tid[1]) jusqu'à long(Tid[1])>0
Pour i de 2 à N Faire
Répéter Ecrire("Tid[",i,"]="),Lire(Tid[i]) jusqu'à (long(Tid[i])>0) et
(Fn existe(Tid[i], 1, i-1,Tid)=Vrai)
Fin Pour

Le Tableau de Déclaration des Nouveaux Types

Type
BacTp = Tableau de 10 Chaines

Le tableau de déclaration des objets globaux

Objet	Type / Nature	Rôle
N	octet	
P	octet	
I	octet	
Tid	BacTp	
Tpw	BacTp	

Id	Chaîne	
Pw	Chaîne	
Existe	Fonction	

Analyse de la fonction Existe

Def Fn Existe (X :Chaîne; d,f: octet; Tid: BacTp) :Entier

Résulat = Existe
❸ Existe ←p
❷ P= [p←0]
Si Tid[i]=X Alors p←i
Fin Si
❶ I=[i←d-1]
Répéter
I←i+1
Jusqu'à (Tid[i]=x) ou (i=f)

Le tableau de déclaration des objets locaux

Objet	Type / Nature	Rôle
P	octet	
I	octet	

La traduction Pascal:

```
Program Authentification;
uses wincrt;
type BacTp=array[1..10]of string;
var Tid,Tpw:BacTp; n,i,p:byte; id,pw:string;
                    {******* la fonction existe *******}
function existe (x:string;d,f:integer;Tid:BacTp):integer;
var i,p:byte;
begin
i:=d-1;
repeat
i:=i+1;
until (Tid[i]=x)or (i=f);
if Tid[i]=x then p:=i else p:=0;
existe:=p;
end;
                    {******* la programme principal *******}
begin
repeat
write('N = ');readln(n);
until n in [2..10];
repeat write('Tid[1] = ');readln(Tid[1]); until length(Tid[1])>0;
for i:= 2 to n do
repeat write('Tid[',i,'] = ');readln(Tid[i]); until (length(Tid[i])>0)
 and (existe(Tid[i],1,i-1,Tid)=0);
repeat write('Tpw[1] = ');readln(Tpw[1]); until length(Tpw[1])>5;
for i:= 2 to n do
repeat write('Tpw[',i,'] = ');readln(Tpw[i]); until length(Tpw[i])>5;
repeat write('id = ');readln(id);until length(id)>0;
```

```
repeat write('pw = ');readln(pw);until length(pw)>5;
p:=existe (id,1,N,tid);
if (p>0)and (Tpw[p]=pw) then write(id,', Bienvenue sur notre site')
                        else write('Vérifiez votre identificateur et/ou votre mot passe');
end.
```

Séance5-Sujet2

A l'approche de la naissance de leur enfant, un couple superstitieux contacte une voyante pour qu'elle lui recommande des lettres **porte-bonheur** pouvant être utilisées dans nomination du futur bébé.

Pour cela, la voyante effectue **N** tirages au hasard de **P** cartes (avec **1≤ P ≤10** et **3 < N < 20**) où chaque carte comporte une lettre majuscule. A chaque tirage en résulte une chaîne de caractères formée par la concaténation des lettres tirées.

Les lettres **porte-bonheur** sont celles les plus tirées dans les différents tirages.

Exemple :

Pour **N = 4** et **P = 5**,

Les tirages effectués ont donné les quatre chaînes de caractères suivantes :

"**HBAMX**", "**MSAIH**", "**MREKA**" et "**DRTYU**".

D'où, les lettres porte-bonheur sont : "**A**" et "**M**" car elles sont les plus tirées (3 fois).

Travail demandé :

Ecrire un programme Pascal qui permet de saisir les deux entiers **N** et **P** et simuler les tâches effectuées par la voyante et d'afficher les lettres **porte-bonheur** correspondantes aux tirages effectués.

Analyse du programme principal

❹ ***Résultat*** = Proc Traitement(R)
❸ R = Formation(N,P,R)
❷ N=[ ]
    Répéter
        Ecrire("N = "), Lire(N)
    Jusqu'à N dans [4..19]
❶ P=[..]
    Répéter
        Ecrire("P = "), Lire(P)
    Jusqu'à P dans [1..10]

Le tableau de déclaration des objets globaux

Objet	Type / Nature	Rôle
N	octet	
P	octet	
R	Chaîne	
Traitement	Procédure	
Formation	Procédure	

Analyse de la procédure Traitement

Def Proc Traitement (R: Chaîne)

❸ ***Résultat*** = Ecrire(W)
❷ W= [W←""]

```
        Pour i de 1 à Long(R) Faire
                Si (Fn Freq(R[i],R)= Fmax) et (pos(R[i],w)=0) Alors W←W+R[i]
                Fin Si
        Fin Pour
❶ Fmax =[Fmax←Fn Freq(R[1],R)]
        Pour i de 2 à Long(R) Faire
                Si Fn Freq(R[i],R)> Fmax Alors Fmax← Fn Freq(R[i],R)
                Fin Si
        Fin Pour
```

Le tableau de déclaration des objets locaux

Objet	Type / Nature	Rôle
I	octet	
Fmax	Octet	
W	Chaîne	
Freq	Fonction	

Analyse de la fonction Freq

Def Fn Freq (C: Caractère; Ch: Chaîne):Octet

Résultat = Freq

```
❷ Freq ← Nb
❶ Nb =[Nb←0]
        Pour i de 1 à Long(Ch) Faire
                Si Ch[i] = C Alors Nb←Nb+1
                Fin Si
        Fin Pour
```

Le tableau de déclaration des objets locaux

Objet	Type / Nature	Rôle
I	octet	
Nb	octet	

Analyse de la procédure Formation

Def Proc Formation (n,p:byte; Var R: Chaîne)

Résultat = R

```
❶ R= [R←""]
        Pour i de 1 à n Faire
                Ch← ""
                Pour j de 1 à p faire
                        Ch←ch+Chr(Aléa(26)+65)
                Fin Pour
                Ecrire(Ch) , R←r+Ch
        Fin Pour
```

Le tableau de déclaration des objets locaux

Objet	Type / Nature	Rôle
I	octet	
J	octet	
Ch	Chaine	

La traduction Pascal:

```
program porte_bonheur;
uses wincrt;
var n,p:byte;  r:string;
                    {******* la procédure formation *******}
procedure Formation(n,p:byte; var r:string);
var i,j:byte; ch:string;
begin
randomize;
r:='';
for i:=1 to n do
begin
ch:='';
for j:=1 to p do   ch:=ch+chr(random(26)+65);
writeln(ch);
r:=r+ch;
end;
end;
                    {******* la fonction freq *******}
function freq(c:char; ch:string):byte;
var nb,i:byte;
begin
nb:=0;
for i:=1 to length(ch) do
if ch[i]=c then nb:=nb+1;
freq:=nb;
end;
                    {******* la procédure traitement *******}
procedure traitement(r:string);
var i,fmax:byte; w:string;
begin
fmax:=freq(r[1],r);
for i:=2 to length(r) do
if freq(r[i],r)>fmax then fmax:=freq(r[i],r);
w:='';
for i:=1 to length(r) do
if (freq(r[i],r)=fmax) and (pos(r[i],w)=0) then w:=w+r[i];
writeln(w);
end;
                    {******* la programme principal *******}
begin
repeat
write('N = '); readln(n);
until n in [4..19];
repeat
write('P =  '); readln(p);
until p in [1..10];
Formation(n,p,r);
Traitement(r);
end.
```

Séance5-Sujet3

Un "**pangramme**" est une chaîne ayant la caractéristique de contenir **toutes** les lettres de l'alphabet (sans distinction entre majuscule et minuscule), même si elle peut parfois être dénuée de sens véritable.

Exemple de chaîne pangramme :

"Monsieur Jack vous dactylographiez bien mieux que votre ami Wolf"

Une chaîne est dite "**palindrome**" si elle se lit de la même façon dans les deux sens de gauche à droite ou de droite à gauche.

Exemple de chaîne palindrome :

"Radar"

Travail demandé :

Ecrire un programme Pascal qui permet de saisir une chaine de caractères non vide formée uniquement par des lettres et des espaces, de déterminer et d'afficher si celle-ci est **pangramme** ou **palindrome** ou **pangramme et palindrome** en même temps.

Analyse du programme principal

❷***Résultat*** = [ ]

```
        Si (Fn Pang(R)) et (Fn Pal(R)) Alors Ecrire("Pangramme et Palindrôme")
              Sinon Si Fn Pang(R) Alors Ecrire("Pangramme")
                    Sinon Si Fn Pal(R) Alors Ecrire("Palindrôme")
                          Sinon Ecrire("Quelconque ")
        Fin Si
```

❶ R = [ ]

```
Répéter
        Ecrire("R = "), Lire(R)
        R← Fn Majuscule(R)
Jusqu'à (Fn Verif(R)=Vrai) et (Long(R)>0)
```

Le tableau de déclaration des objets globaux

Objet	Type / Nature	Rôle
R	Chaîne	
Majuscule	Fonction	
Verif	Fonction	
Pang	Fonction	
Pal	Fonction	

Analyse de la fonction Majuscule

Def Fn Majuscule (R:Chaîne): Chaîne

Résultat = Majuscule

❷ Mjuscule ← Ch

❶ Ch = [Ch←R]

```
        Pour i de 1 à Long(Ch) Faire
              Ch[i]← Majus(Ch[i])
        Fin Pour
```

Le tableau de déclaration des objets locaux

Objet	Type / Nature	Rôle
Ch	Chaîne	
I	Octet	

Analyse de la fonction Verif

Def Fn Verif (R:Chaîne): Booléen

Résultat = Verif
❸ Verif ← test
❷ Test ← R[i] dans ["A".."Z","␣"]
❶ I = [i←0]
Répéter
I←i+1
Jusqu'à (Non(R[i] dans ["A".."Z","␣"])) ou (i=Long(R))

Le tableau de déclaration des objets locaux

Objet	Type / Nature	Rôle
Test	Booléen	
I	Octet	

Analyse de la fonction Panag

Def Fn Pang (R:Chaîne): Booléen

Résultat = Pang
❸ Pang ← test
❷ Test ← Long(Ch) dans [26..27]
❶ Ch = [Ch←R, i←1]
Répéter
Si pos(Ch[i],Ch)<i Alors effacer(ch,i,1)
Sinon i←i+1
Fin Si
Jusqu'à i>Long(Ch)

Le tableau de déclaration des objets locaux

Objet	Type / Nature	Rôle
Test	Booléen	
I	Octet	
Ch	Chaîne	

Analyse de la fonction Pal

Def Fn Pal (R:Chaîne): Booléen

Résultat = Pal
❸ Pal ← test
❷ Test ← R[i] =R[Long(R)-i+1]
❶ I = [tant que pos("␣",R)>0 Faire Efface(R, Pos("␣",R),1)
Fin Tant que, i←0]
Répéter
I←i+1
Jusqu'à (R[i]≠R[Long(R)-i+1]) ou (i=Long(R) div 2)

Le tableau de déclaration des objets locaux

Objet	Type / Nature	Rôle
Test	Booléen	
i	Entier	

La traduction Pascal:

```
Program pang_palindrome;
Uses wincrt ;
var r:string;
                    {******* la fonction Majuscule *******}
function majuscule(r:string):string;
var i:byte; ch:string;
begin
Ch:=R;
for i:=1 to length(Ch) do
Ch[i]:=upcase(Ch[i]);
majuscule:=Ch;
end;
                    {******* la fonction Verif *******}
function verif(r:string):boolean;
var i:byte; test:boolean;
begin
i:=0;
repeat
i:=i+1;
until (not(R[i] in ['A'..'Z',' ']))or (i=length(r));
test:=R[i] in ['A'..'Z',' '];
verif:=test;
end;
                    {******* la fonction Panag *******}
function pang(r:string):boolean;
var i:byte;test :boolean; ch:string;
begin
Ch:=R; i:=1;
repeat
if pos(Ch[i],Ch)<i then delete(Ch,i,1)
else i:=i+1;
until i>length(Ch);
test:=length(Ch) in[26..27];
pang:=test;
end;
                    {******* la fonction Pal *******}
function pal(r:string):boolean;
var i:byte; test:boolean;
begin
while pos(' ',r)>0 do delete(r,pos(' ',r),1);
i:=0;
repeat
i:=i+1;
until (r[i]<>r[length(r)-i+1])or (i=length(r) div 2);
test:= r[i]=r[length(r)-i+1];
pal:=test;
end;
                    {******* le programme principal *******}
begin
```

```
repeat
write('R = ');readln(r);
r:=majuscule(r);
until (verif(r)=true) and (length(r)>0);
if (pang(r)) and(pal(r)) then write('Pangramme et Palindrome')
else if pang(r) then write('Pangramme')
else if pal(r) then write('Palindrome')
else write('Quelconque');
end.
```

Séance5-Sujet4

On se propose de simuler le jeu suivant :

- Le jeu est initialisé par le choix d'un numéro de téléphone à **8 chiffres**, ne commençant pas par zéro et qui est à deviner par un joueur.
- Le joueur propose successivement des chiffres. Pour chaque chiffre proposé, s'il est présent dans une ou plusieurs positions du numéro secret, il sera positionné aux mêmes emplacements.
- À tout moment, si le joueur pense avoir deviné le numéro secret, il peut proposer un numéro. S'il a trouvé le numéro secret, il a gagné.
- Le joueur peut perdre de deux façons : soit il propose un numéro qui n'est pas le bon, soit il a proposé 5 chiffres et n'a toujours pas trouvé le numéro cherché.

Exemple : Ci-dessous un exemple d'exécution pour le numéro secret "**83256221**"

La chaîne de départ à afficher est : "**--------**"

1ère exécution	**2ème exécution**
Proposer un chiffre ? **4** Le numéro de téléphone est :-------- Voulez-vous proposer un numéro ? **N** Proposer un chiffre ? **2** Le numéro de téléphone est :**--2--22-** Voulez-vous proposer un numéro ? **N** Proposer un chiffre ? **3** Le numéro de téléphone est :**-32--22-** Voulez-vous proposer un numéro ? **O** Proposer un numéro : **83256221** Bravo ! Vous avez gagné	Proposer un chiffre ? **7** Le numéro de téléphone est : -------- Voulez-vous proposer un numéro ? **N** Proposer un chiffre ? **2** Le numéro de téléphone est :**--2--22-** Voulez-vous proposer un numéro ? **N** Proposer un chiffre ? **2** Le numéro de téléphone est :**-32--22-** Voulez-vous proposer un numéro ? **N** Proposer un chiffre ? **0** Le numéro de téléphone est :**-32--22-** Voulez-vous proposer un numéro ? **N** Proposer un chiffre ? **5** Le numéro de téléphone est :**-325-22-** Voulez-vous proposer un numéro ? **O** Proposer un numéro : **93256224** Désolé ! Vous avez perdu

Travail demandé :

Ecrire un programme Pascal qui permet de composer un numéro de téléphone (**8 chiffres**) par la concaténation de deux nombres de 4 chiffres choisis aléatoirement par l'ordinateur et simuler le jeu comme décrit ci-dessus.

Analyse du programme principal

❷ ***Résultat*** = [ ]
Si (R=T) ou (Prop=T) Alors Ecrire("Bravo! Vous avez gagné ")
Sinon Ecrire("Désolé! Vous avez perdu ")
Fin Si
❶ R = [R←"--------", K←0, Proc Init(T)]
Répéter
K←K+1
Répéter
Ecrire("Proposer un chiffre. "), Lire(c)
Jusqu'à C dans ["0".."9"]
Proc Remplacer (C, T, R), Ecrire(R)
Répéter
Ecrire("Voulez vous proposer un numéro ?"), Lire(Rep)
Jusqu'à Majus(Rep) dans ["O","N"]
Si Majus(Rep) ="O" Alors Ecrire("Proposer un numéro : ")
Lire(prop)
Fin Si
Jusqu'à (K=5) ou (R=T) ou (Majus(rep)="O")

Le tableau de déclaration des objets globaux

Objet	Type / Nature	Rôle
R	Chaîne	
Prop	Chaine	
T	Constante de valeur="23457632"	
K	Octet	
C	Caractère	
Rep	Caractère	
Init	Procédure	
Remplacer	Procédure	

Analyse de la procédure Init

Def Proc Init (Var T:Chaîne)

Résulat = T
❺ T= ← Ch1+Ch2
❸ Ch1 = Convch(n1, Ch1)
❹ Ch2 [Convch(n2, Ch2)]
Pour i de 1 à 4 – Long(Ch2) Faire
Ch2←"0"+Ch2
Fin Pour
❶ n1← 1000 + Aléa(9000)
❷ n2← Aléa(10000)

Le tableau de déclaration des objets locaux

Objet	Type / Nature	Rôle
Ch1	Chaîne	

Ch2	Chaîne	
n1	Entier	
n2	Entier	

Analyse de la procédure Remplacer

Def Proc Remplacer (C: Caractère; T:Chaîne; Var R:Chaîne)

Résulat = R
❶ R=[ ]
Pour i de 1 à long(T) Faire
Si T[i]=C Alors R[i]←C
Fin Si
Fin Pour

Le tableau de déclaration des objets locaux

Objet	Type / Nature	Rôle
I	Octet	

La traduction Pascal:

```
program jeu;
var R,Prop,T :string; k:byte; c,rep:char;
                              {******* la procédure Init *******}
procedure Init(Var T:string);
var i,n1,n2:integer;    ch1,ch2:string;
begin
randomize;
 n1:=1000+random(9000);
n2:=random(10000);
str(n1,ch1);
str(n2,ch2);
for i:=1 to 4- length(ch2)  do
        ch2:='0'+ch2;
T:=ch1+ch2;
end;
                           {******* la procédure Remplacer *******}
procedure Remplacer(c:char;T:string; var R:string);
var i:byte;
begin
for i:=1 to length(T) do
if t[i]=c then R[i]:=c;
end;
                           {******* le programme principal *******}
begin
R:='--------'; k:=0;Init(T);
repeat
k:=k+1;
repeat
write('Proposer un chiffre. ');readln(c);
until c in ['0'..'9'];
Remplacer(c,T,R);writeln(R);
```

```
repeat
write('Voulez-vous proposer un numéro ? ');readln(Rep);
until upcase(rep) in ['O','N'];
if upcase(rep)='O' then begin
                write('Propser un numéro :');
                readln(prop);
                end;
until (k=5) or (R=T)or(upcase(rep)='O');
if (R=T) OR (Prop=T) then write('Bravo ! Vous avez gagné')
else write('Désolé! Vous avez perdu');
end.
```

Séance6-Sujet1

La suite de Frank est définie comme suit :

$$\begin{cases} U_1 = x \\ U_n = U_{n-1} + PGCD(N, U_{n-1}), \text{ pour tout } n \geq 2 \end{cases}$$

Soit la suite **V** définie en fonction de la suite de Frank, comme suit :

$V_n = U_n - U_{n-1}$ pour tout $n \geq 2$

Les termes de la suite **V** sont soit égale à **1**, soit un nombre premier. Après le calcul d'un certain nombre de terme, la suite **V** est dite équilibrée si et seulement si le nombre des **1** est égal à celui des entiers premiers.

Exemples :

- Pour **x = 4**, le calcul des termes de la suite **V** donne : V_2=2, V_3=3, V_4=1, V_5=5, V_6=3, V_7=1, V_8=l, V_9=l. ➔Cette suite est équilibrée car le nombre des 1 est égal au nombre des entiers premiers.
  ➔Le programme affiche : "**La suite V est équilibrée après le calcul de 8 termes**"
- Pour **x = 7**, le calcul des 30 premiers termes (de V_2 à V_{31}) de la suite V donne :
  1, 1, 1, 5, 3, 1, 1, 1, 1, 11,3, 1, l, 1, 1, 1, 1, l, 1, 1, 1, 23, 3, 1, 1, 1, 1, 1, 1, 1
  ➔Cette suite n'est pas équilibrée car le nombre des 1 est différent du nombre des entiers premiers après le calcul de 30 termes de la suite **V**.
  ➔Le programme affiche : "**Impossible d'atteindre l'équilibre après le calcul de 30 termes**"

Travail demandé :

Ecrire un programme Pascal qui permet de saisir le premier terme **x** ($2 \leq x \leq 10$) de la suite **U,** de calculer et d'afficher le rang à partir duquel la suite **V** est équilibrée. Dans le cas où on calcule **30** termes et que l'équilibre ne soit pas atteint on affiche le message « **Impossible d'atteindre l'équilibre après le calcul de 30 termes** »

NB : 1) Soient **a** et **b** deux entiers et **r** le reste de la division euclidienne de **a** par **b**. Le **PGCD(a,b)**=**PGCD(b,r)** jusqu'à **r** = 0. Le **PGCD(a,b)** est égal au dernier reste non nul.

2) Ne pas vérifier que les termes différents de **1** de la suite **V** sont premiers.

Analyse du programme principal

❸ ***Résultat*** = [ ]

Si nb1=nbp Alors Ecrire("La suite V est équilibrée après le calcul de ", 2*nb1," termes")
Sinon Ecrire ("Impossible d'atteindre l'équilibre après le calcul de 30 termes ")
Fin Si

❷ (nb1,nbp) = Proc Suites(x,nb1,nbp)

❶ X = [ ]

Répéter
Ecrire ("X = "), Lire(x)
Jusqu'à x dans [2..10]

Le tableau de déclaration des objets globaux

Objet	Type / Nature	Rôle
X	Octet	
Nb1	Octet	
Nbp	Octet	
Suites	Procédure	

Analyse de la procédure Suites

Def Proc Suites (X: octet; Var Nb1,Nbp:octet)

Résulat = Nb1, Nbp
❶ (Nb1, Nbp) = [nb1←0, nbp←0, U←X, n←1]
Répéter
N←n+1
Up←u
U← Up+ Fn PGCD(n, Up)
V← U- Up
Si V=1 Alors Nb1←Nb1+1
Sinon Nbp← Nbp+1
Fin Si
Jusqu'à (nbp=nb1) ou (n=31)

Le tableau de déclaration des objets locaux

Objet	Type / Nature	Rôle
U	Octet	
Up	Octet	
N	Octet	
V	Octet	
PGCD	Fonction	

Analyse de la fonction PGCD

Def Fn PGCD (a,b :Entier) :Entier

Résulat = PGCD
❷ PGCD ← V
❶ V =[V←a; W←b]
Tant que W>0 Faire
r← V mod W
V ← W
W ←r
Fin Tant que

Le tableau de déclaration des objets locaux

Objet	Type / Nature	Rôle
V	Entier	
W	Entier	
R	Entier	

La traduction Pascal:

```
Program Frank;
uses wincrt;
var   x,nb1,nbp:byte;
                    {******* la fonction PGCD *******}
function PGCD(a,b:integer):integer;
var v,w,r:integer;
begin
v:=a; w:=b;
while w>0 do begin
r:= v mod w;
v:=w;
w:=r;
end;
PGCD:=v;
end;
                    {******* la procédure Suites *******}
Procedure Suites(x:byte; var nb1,nbp:byte);
var U,Up,v,n:byte;
begin
nb1:=0;nbp:=0;u:=x;n:=1;
repeat
n:=n+1;
up:=u;
u:=up+PGCD(n,up);
v:=u-up;
 if v=1 then nb1:=nb1+1
     else nbp:=nbp+1;
until (nbp=nb1) or (n=31);
end;
                    {******* le programme principal *******}
begin
repeat
Write('X = '); readln(x);
until x in [2..10];
Suites(x,nb1,nbp);
if nb1=nbp then write('La suite V est équilibrée aprés le calcul de ',2*nb1,' termes')
else write('impossible d''atteindre l''équilibre de la suite V après le calcul de ses 30
premiers termes ');
end.
```

Séance6-Sujet2

N est un nombre de **SIERPINSKI** s'il vérifie la propriété suivante : $\mathbf{N = K * 2^{i} + 1}$ avec **i** un entier strictement positif.

Exemple :

Pour **K = 3** et pour tout **i** variant de **1** à **10**, Le programme affiche tous les nombres de **SIERPINSKI** ci-dessous :

K	i	N	
3	1	7	Premier
3	2	13	Premier
3	3	25	
3	4	49	
3	5	97	Premier
3	6	193	Premier
3	7	385	
3	8	769	Premier
3	9	1537	
3	10	3073	

N.B. :

Un nombre est dit premier s'il n'est divisible que par 1 et par lui-même. Par définition, 1 n'est pas premier.

Travail demandé :

Ecrire un programme Pascal qui permet de choisir aléatoirement un entier **K** de l'intervalle **[1, 5]**, de chercher et d'afficher tous les nombres de **SIERPINSKI** en variant **i** de **1** à **10**, tout en mentionnant le terme "**Premier**" devant les nombres de **SIERPINSKI** qui sont premiers comme indiqué dans l 'exemple ci-dessus.

Analyse du programme principal

❷ ***Résultat*** = [P←1]

```
    Pour i de 1 à 10 Faire
        P←P*2
        N← K*P+1
        Ecrire(K,"    ", i,"    ", n,"     ",Fn Premier(n))
    Fin Pour
```

❶ K← Aléa (5) +1

Le tableau de déclaration des objets globaux

Objet	Type / Nature	Rôle
P	Entier	
I	Octet	
N	Entier	
K	Octet	
Premier	Fonction	

Analyse de la fonction Premier

Def Fn Premier (K :Entier) :Chaine

Résultat = Premier

❶ Premier ← R

❷ R = [ ]

```
    Si K<2 Alors R ← ""
    Sinon si K=2 Alors R ← "Premier"
    Sinon si K mod 2 = 0 Alors R ← ""
    Sinon
        i←1
        Répéter
            i←i+2
        Jusqu'à (i>Racinecarré(K) ou (K mod i =0)
```

Si(i>Racinecarré(K)) Alors R←"Premier"
Sinon R ←""
Fin Si

Le tableau de déclaration des objets locaux

Objet	Type / Nature	Rôle
I	Entier	
R	Chaine[7]	

La traduction Pascal:

```
program sierpinski;
var n,p:integer; i,k:byte;
                    {******* la fonction Premier *******}
function Premier(k:integer):string;
var i:integer;
R:string[7];
begin
if k<2 then R:=''
else if k=2 then R:='Premier'
else if k mod 2=0 then R:=''
else begin
i:=1;
repeat
i:=i+2;
until (k mod i=0)or(i>sqrt(k));
if i>sqrt(k) then R:='Premier' else R:='';
end;
Premier:=R;
end;
                    {******* le programme principal *******}
begin
randomize();
k:=random(5)+1;
p:=1;
for i:=1 to 10 do begin
        p:=p*2;
        n:=k*p+1;
        writeln(k:4,i:4,n:7,Premier(n));
end;
end.
```

Séance6-Sujet3

Un nombre N de deux chiffres peut être affiché sous la forme de suite de sommes d'entiers consécutifs.

Exemple :

Les sommes consécutives de **N= 21** sont :

21 = 1+2+3+4+5+6
21 = 6+7+8
21 = 10+11

NB : un nombre de deux chiffres peut avoir de 0 à 5 suites dentiers consécutifs.

Travail demandé :

Ecrire un programme Pascal qui permet de saisir deux entiers **N** et **M** positifs de deux chiffres, de déterminer celui qui a le plus de suites de sommes d'entiers consécutifs et de l'afficher ainsi que ses suites.

Exemple : Pour N=12 et M = 54

N a une seule suite dentiers consécutifs qui est 12 = 3+4+5 et **M** a 3 suites donc le programme affiche :

54 et ses suites d'entiers consécutifs sont :

54 = 2+3+4+5+6+7+8+9+10
54 = 12+13+14+15
54 = 17+18+19

Analyse du programme principal

❺***Résultat*** = [ ]
Si Nsn> Nsm Alors Proc Affiche(Sn, Nsn, N)
Sinon Proc Affiche(Sm, Nsm, M)
Fin Si
❸ (Nsn, Sn) = Proc Suites(n, Sn, Nsn)
❹(Nsm, Sm) = Proc Suites(m, Sm, Nsm)
❶ n = Proc Saisie(n)
❷ m = Proc Saisie(m)

Le Tableau de Déclaration des Nouveaux Types

Type
Tab = Tableau de 5 octets

Le tableau de déclaration des objets globaux

Objet	Type / Nature	Rôle
N	Octet	
M	Octet	
Nsn	Octet	
Nsm	Octet	
Sn	Tab	
Sm	Tab	
Suites	Procédure	
Affiche	Procédure	
Saisie	Procédure	

Analyse de la procédure Saisie

Def Proc Saisie(Var K:Octet)

Résultat = K
❶ K = [ ]
Répéter
Ecrire("Saisir un entier "), Lire(K)
Jusqu'à K dans [10..99]

Analyse de la procédure Suites

Def Proc Suites(N :Octet; Var T: Tab; Var K: Octet)

Résultat = T, K
❶ (T,K) = [K←0]
Pour i de 1 à N div 2 Faire
J←i
S←j
Répéter
J←j+1
S←S+j
Jusqu'à S>= N
Si S=N Alors K←K+1, T[K]←i
Fin Si
Fin Pour

Le tableau de déclaration des objets locaux

Objet	Type / Nature	Rôle
I	Octet	
J	Octet	
S	Octet	

Analyse de la procedure Affiche

Def Proc Affiche (T:Tab; K,N :Octet)

❶ ***Résultat*** = Ecrire(N, " et ses suites d'entiers consécutifs sont : ")
Pour i de 1 à K Faire
J←T[i]
S←T[i]
Ecrire(N, " = ",T[i])
Répéter
J←j+1
S←S+j
Ecrire(" + ",j)
Jusqu'à S= N
Fin Pour

Le tableau de déclaration des objets locaux

Objet	Type / Nature	Rôle
S	Octet	
I	Octet	
J	Octet	

La traduction Pascal:

```
Program partition;
type tab =array[1..5] of byte;
var sn,sm:tab;
var n,m,nsn,nsm:byte;
                    {******* la procédure Suites *******}
Procedure Suites(n:byte; Var t:tab;var k:byte);
```

```
var i,j,s:byte;
begin
k:=0;
for i:=1 to n div 2 do begin
                        j:=i;s:=j;
                        repeat
                        j:=j+1;
                        s:=s+j;
                        until s>=n;
                        if s=n then begin k:=k+1;t[k]:=i; end;
                        end;
end;
                              {******* la procédure Affiche *******}
Procedure affiche(t:tab; k:byte; n:byte);
var i,j,s:byte;
begin
for i:=1 to k do begin
        s:=t[i];
        j:=T[i];
        write(n ,' = ',s);
                repeat
                j:=j+1;
                s:=s+j;
                write('+',j);
                until s=n; writeln;
        end;
end;
                              {******* la procédure Saisie *******}
Procedure Saisie( Var K:byte);
begin
repeat
write('Saisir un entier : ');readln(k);
until k in [10..99];
end;
                              {******* le programme principal *******}
begin
Saisie(N);
Saisie(M);
Suites(n,Sn,nsn);
Suites(m,Sm,nsm);
if nsn > nsm then affiche(Sn, nsn,n) else affiche(sm, nsm,m);
end.
```

Séance6-Sujet4

On se propose de construire à partir d'un chiffre **E** impair donné une pyramide composée de **L** lignes. Chaque ligne est calculée en fonction de la ligne qui la précède en insérant à son début et à sa fin un chiffre **C** tel que :

C = (la somme des chiffres de la ligne précédente + nombre de chiffres de la ligne précédente) MOD 10.

La dernière ligne de la pyramide correspond au premier nombre divisible par **7**.

Exemple : Pour E = 1

1
212
82128
6821286
068212860
20682128602
8206821286028
682068212860286 {***C'est le premier nombre divisible par 7***}

La ligne 6 est calculée en insérant le chiffre C au début et à la fin du nombre de la ligne 5.
Avec C = ((0+6+8+2+1+2+8+6+0)+9) MOD 10= 42 MOD 10 = 2
NB : le chiffre 0 à gauche est pris en compte dans le calcul du nombre de chiffres de la ligne précédente

Pour déterminer si un nombre **N** est divisible par **7**, il suffit de le décomposer en des tranches de trois chiffres en commençant par la droite et d'insérer alternativement des **+** et des **-** devant les tranches en commençant par l'opérateur +. On effectue l'opération ainsi écrite, si le ***Résultat*** est divisible par **7** alors N est divisible par 7.

Exemple :

Pour N =682068212860286 et en appliquant la règle de divisibilité par **7** ci-dessus, on obtient +286-860+212-068+682 = 252 qui est divisible par 7 donc N est divisible par 7.

Travail Demandé

Ecrire un programme Pascal qui permet de saisir un entier **E** impair (1≤E≤9), d'afficher les entiers correspondants à **E** selon le principe décrit précédemment à raison d'un entier par ligne.

N.B : Le candidat n'est pas appelé à afficher les entiers sous la forme d'une pyramide

Analyse du programme principal

❷ ***Résultat*** = Proc Traitement (E)
❶ E = [ ]
Répéter
Ecrire("E = "), Lire(E)
Jusqu'à E dans [1, 3, 5, 7, 9]

Le tableau de déclaration des objets globaux

Objet	Type / Nature	Rôle
E	Octet	
Traitement	Procédure	

Analyse de la procedure Traitement

Def Proc Traitement (N :Octet)

❶ ***Résultat*** = [L←Chr(N+48), Ecrire(L)]
Tant que Fn DivPar7(L) = Faux Faire
C←(Fn SommeChiff(L)+Long(L)) mod 10
L← Chr(C+48) + L + Chr(C+48)
Ecrire(L)
Fin Tant que

Le tableau de déclaration des objets locaux

Objet	Type / Nature	Rôle
L	Chaine	
DivPar7	Fonction	
C	Octet	
SommeChiff	Fonction	

Analyse de la fonction SommeChiff

Def Fn SommeChiff (K:Chaine) :Entier

Résultat = SommeChiff
❷ SommeChiff← S
❶ S = [S← 0]
Pour i de 1 à Long(k) Faire
S← S +ord(k[i])-48
Fin Pour

Le tableau de déclaration des objets locaux

Objet	Type / Nature	Rôle
I	Octet	
S	Entier	

Analyse de la fonction DivPar7

Def Fn DivPar7 (K:Chaine) :Boolean

Résultat = DivPar7
❸DivPar7← test
❷Test← S mod 7 = 0
❶ S = [S←0, signe←0,
Si Long(K) mod 3 =1 Alors K←"00"+K
Sinon Si long(k) mod 3 =2 Alors K←"0"+K
Fin si]
Répéter
Signe ← signe+1
Valeur(Souschaine(K, Long(K)-2),3),N,er)
Efface(K,Long(K)-2,3)
Si signe mod 2 = 0 Alors S← S-n
Sinon S←S+n
Fin Si
Jusqu'à Long(K)=0

Le tableau de déclaration des objets locaux

Objet	Type / Nature	Rôle
Signe	Octet	
S	Entier	
N	Entier	
Er	Entier	
Test	Booléen	

La traduction Pascal:

```
Program pyramide;
var E:byte;
                    {******* la fonction DivPar7 *******}
function DivPar7(K:string):boolean;
var signe: Byte; s,N,er:integer;
test:boolean;
begin
```

```
s:=0; signe:=0;
if length(k) mod 3 =1 then k:='00'+k
   else if length(k) mod 3 =2 then k:='0'+k;
repeat
signe:=signe+1;
val(copy(k,length(k)-2,3),N,er);
delete(k,length(k)-2,3);
if signe mod 2 = 0 then s:=s-n else s:=s+n;
until length(k)=0;
test:= s mod 7 =0;
divpar7:= test;
end;
```

{**** la fonction Sommechiff *******}***

```
function Sommechiff(K:string):integer;
var s:integer; i:byte;
begin
s:=0;
for i:=1 to length(k) do s:=s+ord(k[i])-48;
Sommechiff:=s;
end;
```

{**** la procédure Traitement *******}***

```
procedure Traitement(N:byte);
var  L :string; c:byte;
begin
L:=chr(N+48); writeln(L);
while (DivPar7(L)=false)  do begin
        c:=(Sommechiff(L)+length(L)) mod 10;
        L:=Chr(C+48)+L+Chr(C+48); Writeln(L);  end;
end;
```

{**** le programme principal *******}***

```
begin
repeat
write('E = '); readln(E);
until E in [1,3,5,7,9];
Traitement(E);  end.
```