

Exercice 1 (08 pts) : On considère le code (choisir entre l'algorithme et le programme) suivant manipulant trois matrices d'entiers A, B et C ayant n lignes et m colonnes :

Algorithme	Programme
<pre> coef ← 1; Pour j ← 1 jusqua m faire Pour i ← 1 jusqua n faire coef ← 1 - coef ; B[i,j] ← A[i,j] ; A[i,j] ← coef * A[i,j] ; C[i,j] ← B[i,j] - A[i,j] ; fpour ; fpour ; </pre>	<pre> coef = 1; for(j=0;j< m;j++){ for(i=0;i< n;i++){ coef = 1 - coef; B[i][j] = A[i][j]; A[i][j] = coef * A[i][j]; C[i][j] = B[i][j] - A[i][j]; } } </pre>

1. Le contenu initial de B (i.e. avant les boucles) va-t-il influer (هل يؤثر) sur son contenu à la fin du code ? Justifier.

2. En supposant que $A = \begin{pmatrix} 2 & 7 & 5 & 3 \\ 7 & 1 & 3 & 4 \\ 4 & 8 & 9 & 5 \end{pmatrix}$ au début du code, quel sera le contenu de A, B et C à la

fin de l'exécution de celui-ci.

3. Que fait cet algorithme (ou ce programme) ?

Exercice 2 (12 pts) : On considère un tableau d'entiers **vect** de taille n ($n \leq 100$) et un intervalle $I=[a,b]$ avec a et b entiers ($a < b$). Une **tranche** d'un tableau est une suite d'éléments consécutifs de celui-ci. La longueur d'une tranche est le nombre de ses éléments.

Écrire un algorithme (ou un programme C) qui affiche :

1. La tranche de **vect** la plus longue et dont les éléments sont tous dans I et
2. Sa longueur.

Exemples : Voici 3 exemples de vecteurs **vect**. Pour $I=[3,7]$ et

vect= <table border="1" style="display: inline-table; vertical-align: middle; border-collapse: collapse; width: 150px; height: 20px;"> <tr><td>5</td><td>8</td><td>3</td><td>4</td><td>1</td><td>7</td><td>7</td><td>6</td><td>1</td></tr> </table>	5	8	3	4	1	7	7	6	1	Il y a 3 tranches d'éléments de I dans vect. La plus longue est (7,7,6) et sa longueur est 3. vect= <table border="1" style="display: inline-table; vertical-align: middle; border-collapse: collapse; width: 150px; height: 20px;"> <tr><td>5</td><td>6</td><td>3</td><td>6</td><td>2</td><td>1</td><td>7</td><td>5</td><td>1</td></tr> </table> La tranche (5,6,3,6) est la plus longue. Sa longueur est 4. vect= <table border="1" style="display: inline-table; vertical-align: middle; border-collapse: collapse; width: 150px; height: 20px;"> <tr><td>2</td><td>1</td><td>8</td><td>1</td><td>8</td><td>8</td><td>9</td><td>2</td><td>1</td></tr> </table> Aucune tranche de I. La longueur est 0.	5	6	3	6	2	1	7	5	1	2	1	8	1	8	8	9	2	1
5	8	3	4	1	7	7	6	1																				
5	6	3	6	2	1	7	5	1																				
2	1	8	1	8	8	9	2	1																				

Remarque : Si **vect** contient plus d'une tranche de longueur maximale, afficher une seule suffit.

Remarques générales :

Utiliser soit le langage algorithmique que vous avez appris au cours, soit le langage C. Le mélange dans un même exercice n'est pas accepté. L'utilisation du portable et/ou de la machine à calculer est **interdite**. Il est **interdit** aussi de se prêter quoi que ce soit (effaceur, stylo, ...) entre étudiants.

```

/************* Exercice 1 *****/
/************* Exercice 1 *****/
/*Q1: Non car les valeurs de B sont écrasées par celles de ----- Q1: 2.5 pts
   A avant qu'elles ne soient utilisées */
/*Q2: On obtient ----- Q2: 3 pts
A: 0 7 0 3
   7 0 3 0
   0 8 0 5
B: 2 7 5 3
   7 1 3 4
   4 8 9 5
C: 2 0 5 0
   0 1 0 4
   4 0 9 0
*/
/*Q3: Ce code sauvegarde la matrice A dans B et crée deux ----- Q3: 2.5 pts
   matrices A et C contenant les valeurs initiales de A
   en annulant les position en damier de chacune d'elles
   de manière complémentaire */
/************* Exercice 2 *****/
/************* Exercice 2 *****/
#include <stdio.h>
int vect[100],n,i,seqLen,maxSeqLen,a,b,last;
int main(){
printf("\nDonner la taille de votre tableau(<=100):"); //----- lecture: 2 pts
scanf("%d",&n);
for(i=0;i<n;i++){
    printf("\nDonner l'element %d:",i+1);
    scanf("%d",&vect[i]);
}
printf("\nDonner la valeur de début de l'intervalle I:");
scanf("%d",&a);
printf("\nDonner la valeur de fin de l'intervalle I:");
scanf("%d",&b);

//Plus longue tranche de vect dans [a,b] ----- traitements: 7 pts
maxSeqLen=seqLen=0;
for(i=0;i<n;i++){
    if((vect[i]>=a) && (vect[i]<=b)){
        seqLen++;
    }else{
        if(seqLen>maxSeqLen){
            maxSeqLen=seqLen;
            last=i-1;
        }
        seqLen=0;
    }
}
if((vect[n-1]>=a) && (vect[n-1]<=b)&&(seqLen>maxSeqLen)){
    maxSeqLen=seqLen;
    last=n-1;
}
if(maxSeqLen){ //----- affichage: 3 pts
    printf("\n La plus longue tranche mesure %d.",maxSeqLen);
    printf("\n La tranche est:(");
    for(i=last-maxSeqLen+1;i<=last;i++){
        printf("\t%d",vect[i]);
    }
    printf(")");
}else{
    printf("\n Aucune tranche du vecteur dans [%d,%d], longueur 0",a,b);
}

return 0;
}

```

Exercice (20 pts) :

Nous avons comme données :

- deux matrices A et B d'entiers de mêmes dimensions $n \times m$ (n et $m \leq 10$) et
- une seule position $[k,l]$ dans ces deux matrices ($[k,l]$ est une position valide).

Par l'élément $[k,l]$ passe une seule diagonale (parallèle à la diagonale principale) qui décompose chaque matrice en deux parties, une inférieure à la diagonale et l'autre supérieure. On associe les éléments de cette diagonale à la partie supérieure.

Écrire un algorithme qui, sans utiliser une autre matrice, permute entre A et B les parties ayant le moins d'éléments (en cas d'égalité permuter les parties inférieures).

Exemple 1: Pour $[k,l]=[3,1]$ et

$$A = \begin{vmatrix} 2 & 9 & 7 \\ 1 & 4 & -3 \\ 5 & 8 & 5 \\ 3 & -2 & 6 \\ 7 & 1 & 4 \end{vmatrix} \text{ et } B = \begin{vmatrix} 5 & 3 & 7 \\ 1 & -7 & 9 \\ 2 & -5 & 2 \\ 7 & 2 & 4 \\ 8 & -9 & 3 \end{vmatrix}$$

Les parties inférieures ont le moins d'éléments (3 contre 12). Donc après permutation :

$$A = \begin{vmatrix} 2 & 9 & 7 \\ 1 & 4 & -3 \\ 5 & 8 & 5 \\ 7 & -2 & 6 \\ 8 & -9 & 4 \end{vmatrix} \text{ et } B = \begin{vmatrix} 5 & 3 & 7 \\ 1 & -7 & 9 \\ 2 & -5 & 2 \\ 3 & 2 & 4 \\ 7 & 1 & 3 \end{vmatrix}$$

Exemple 2: Pour $[k,l]=[1,2]$ et

$$A = \begin{vmatrix} 2 & 9 & 5 \\ 1 & 4 & -3 \\ 5 & 8 & 5 \\ 3 & -2 & 6 \\ 7 & 1 & 4 \\ 2 & 9 & 1 \\ 5 & 9 & 0 \end{vmatrix} \text{ et } B = \begin{vmatrix} 5 & 3 & 7 \\ 1 & -7 & 9 \\ 2 & -5 & 2 \\ 7 & 2 & 4 \\ 8 & -9 & 3 \\ 3 & 1 & 6 \\ -5 & 2 & 1 \end{vmatrix}$$

Les parties supérieures ont le moins d'éléments (3 contre 18). Donc après permutation :

$$A = \begin{vmatrix} 2 & 3 & 7 \\ 1 & 4 & 9 \\ 5 & 8 & 5 \\ 3 & -2 & 6 \\ 7 & 1 & 4 \\ 2 & 9 & 1 \\ 5 & 9 & 0 \end{vmatrix} \text{ et } B = \begin{vmatrix} 5 & 9 & 5 \\ 1 & -7 & -3 \\ 2 & -5 & 2 \\ 7 & 2 & 4 \\ 8 & -9 & 3 \\ 3 & 1 & 6 \\ -5 & 2 & 1 \end{vmatrix}$$

Exemple 3: Pour $[k,l]=[n,1]$, aucune permutation ne sera réalisée, car la partie inférieure est la plus petite, mais elle est vide.

Remarque générale :

L'utilisation du portable et/ou de la machine à calculer est **interdite**. Il est **interdit** aussi de se prêter quoi que ce soit (effaceur, stylo, ...) entre étudiants.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int A[10][10],B[10][10];
int n,m,i,j,nbrElemAll,nbrElemSup,k,l,tmp;
int main(){
    //Info matrices
    printf("\n Donner n et m de la matrice (<10):");
    scanf("%d %d",&n,&m);
    //Au lieu de lire j'utilise des val aleatoires
    srand(time(NULL));
    for(i=0;i<n;i++)for(j=0;j<m;j++)A[i][j]=rand()%10;
    for(i=0;i<n;i++)for(j=0;j<m;j++)B[i][j]=rand()%10;

    printf("\n Donner la position k (de 0 et %d) et l (de 0 et %d):",n-1,m-1);
    scanf("%d %d",&k,&l);

    printf("A :\n");
    for(i=0;i<n;i++){
        for(j=0;j<m;j++)
            printf("%3d",A[i][j]);
        printf("\n");
    }
    printf("B :\n");
    for(i=0;i<n;i++){
        for(j=0;j<m;j++)
            printf("%3d",B[i][j]);
        printf("\n");
    }

    nbrElemAll=n*m;
    nbrElemSup=0;
    for(i=0;i<m;i++)for(j=0;j<m;j++)if(i-j<=k-l)nbrElemSup++;
    printf("\n Nbr elem sup %d et inf %d",nbrElemSup,nbrElemAll-nbrElemSup);//en plus

    if(nbrElemSup<nbrElemAll-nbrElemSup){
        for(i=0;i<n;i++)
            for(j=0;j<m;j++)
                if(i-j<=k-l){
                    tmp=A[i][j];A[i][j]=B[i][j];B[i][j]=tmp;
                }
    }else{
        for(i=0;i<n;i++)
            for(j=0;j<m;j++)
                if(i-j>k-l){
                    tmp=A[i][j];A[i][j]=B[i][j];B[i][j]=tmp;
                };
    }
    printf("Apres permutation :\n");

    printf("A :\n");
    for(i=0;i<n;i++){
        for(j=0;j<m;j++)
            printf("%3d",A[i][j]);
        printf("\n");
    }
    printf("B :\n");
    for(i=0;i<n;i++){
        for(j=0;j<m;j++)
            printf("%3d",B[i][j]);
        printf("\n");
    }
}
```

Exercice 1 (5 pts) : Soit l'algorithme suivant :

Algorithme sahlbfz ;

 var n,a,b,c,i;

debut

 ecrire(' Donner un entier SVP ') ;

 lire(n) ;

 a ← 1 ; b ← a ; c ← b ;

 pour i ← 1 jusqu'à n faire

 a ← a * i ;

 b ← b + 1 ;

 c ← c * 2 ;

 fpour ;

 ecrire(' a = ', a, ' b = ', b, ' c = ', c) ;

fin.

1. Dérouler l'algorithme pour n = 4.

2. Déduire ce qu'il fait.

3. Traduire l'algorithme en C.

Exercice 2 (4 pts) :

Écrire un algorithme ou un programme C qui détermine, sans utiliser de tableaux, l'écart entre la valeur maximale et la valeur minimale d'une liste d'entiers de taille ≤ 100 .

Exercice 3 (5 pts) :

Écrire un algorithme ou un programme C qui réalise un remplissage **basculé-centré** d'un tableau Tab avec n valeurs entières (n doit être impair et ≤ 100).

Un remplissage basculé-centré consiste à mettre la première valeur lue au milieu de Tab, puis à remplir le reste de façon alternée droite-gauche, droite-gauche jusqu'à remplissage total du tableau.

Exemple : Pour les n=7 valeurs lues 2 → 4 → 6 → 3 → 10 → 8 → 12, le tableau Tab contiendra à la fin (12, 10, 6, 2, 4, 3, 8). C'est-à-dire, la première valeur introduite 2 sera placée au milieu, puis la seconde valeur 4 sera placée à droite de la valeur 2. Ensuite, la troisième valeur 6 sera placée à gauche de 2, puis 3 à droite du 4, puis 10 à gauche du 6, et ainsi de suite jusqu'à remplissage des n=7 valeurs.

Exercice 4 (6 pts) :

Écrire un algorithme ou un programme C qui met les valeurs d'un tableau Vect d'entiers de

taille n (≤ 100) dans une matrice A de taille nl \times nc (avec n = nl \times nc) à partir d'une position donnée (k,l) de A.

Le remplissage de A par les éléments de Vect doit respecter l'ordre de celui-ci. C'est-à-dire, après remplissage, si on parcourt A ligne par ligne à partir de la case (k,l) jusqu'à la fin, puis on continue à partir de la première case jusqu'à la case qui précède la case (k,l), on retrouve les éléments de Vect et dans le même ordre.

Exemple : Pour Vect=(10, 4, 7, 12, 9, 5, 8, 7, 5, 3, 6, 1), A une matrice de nl=3 lignes et nc=4 colonnes et la position de début de remplissage (k,l)=(2,3), cet algorithme doit générer la matrice A suivante :

$$A = \begin{pmatrix} 8 & 7 & 5 & 3 \\ 6 & 1 & \mathbf{10} & 4 \\ 7 & 12 & 9 & 5 \end{pmatrix}$$

Car si on parcourt A ligne par ligne à partir de la case (2,3) on trouve les valeurs 10 \rightarrow 4 \rightarrow 7 \rightarrow 12 \rightarrow 9 \rightarrow 5, puis en continuant depuis le début \rightarrow 8 \rightarrow 7 \rightarrow 5 \rightarrow 3 \rightarrow 6 \rightarrow 1, on retrouve les éléments de Vect.

Remarque générale :

L'utilisation du portable et/ou de la machine à calculer est **interdite**.

Il est **interdit** aussi de se prêter quoi que ce soit (effaceur, stylo, ...) entre étudiants.

```

1 //*****
2 //***** Exercice 1 : 5 pts *****
3 //*****
4 //Q1: déroulement -----> (1.5 pt)
5 /*
6 |instr.      n   a   b   c   i|
7 -----
8 |lire(n)      4   ?   ?   ?   ?|
9 |a <- 1       4   1   ?   ?   ?|
10|b <- a       4   1   1   ?   ?|
11|c <- b       4   1   1   1   ?|
12|i <- 1       4   1   1   1   1|
13|a <- a * i   4   1   1   1   1|
14|b <- b + 1   4   1   2   1   1|
15|c <- c * 2   4   1   2   2   1|
16|i <- i+1     4   1   2   2   2|
17|a <- a * i   4   2   2   2   2|
18|b <- b + 1   4   2   3   2   2|
19|c <- c * 2   4   2   3   4   2|
20|i <- i+1     4   2   3   4   3|
21|a <- a * i   4   6   3   4   3|
22|b <- b + 1   4   6   4   4   3|
23|c <- c * 2   4   6   4   8   3|
24|i <- i+1     4   6   4   8   4|
25|a <- a * i   4   24  4   8   4|
26|b <- b + 1   4   24  5   8   4|
27|c <- c * 2   4   24  5   16  4|
28|i <- i+1     4   24  5   16  5|
29----- */
30//Q2: rôle : a -> factorielle de n,          (0.5 pt)
31//           b -> n+1                      (0.5 pt)
32//           c -> 2 à la puissance n.        (0.5 pt)
33//Q3: traduction -----> (2 pts)
34#include <stdio.h>
35int n,a,b,c,i;
36
37int main(){
38    printf("Donner un entier SVP :");
39    scanf("%d",&n);
40    a = 1; b = a; c = b;
41    for(i=1;i<=n;i++){
42        a = a * i;
43        b = b + 1;
44        c = c * 2;
45    }
46    printf("\n a = %d b = %d c = %d",a,b,c);
47    return 0;
48}
49//*****
50//***** Exercice 2 : 4 pts *****
51//*****
52#include <stdio.h>
53int a,max,min,i,n;
54
55int main(){
56    printf("Donner la taille de la liste <= 10 :");//lecture des inputs -----> 0.5
57    pt
58    scanf("%d",&n);//
59    printf("\n Donner le premier élément://");
60    min=max=a;//-----
61    for(i=1;i<n;i++){//boucle -----> 1 pt
62        printf("\n Donner l'élément n°%d : ",i+1);
63        scanf("%d",&a);//-----
64        if(a<min)min=a;//calcul du min:initialisation 0.5, cond 0.5 pt -----> 1 pt
65        if(a>max)max=a;//calcul du max:initialisation 0.5, cond 0.5 pt -----> 1 pt
66    }
67    printf("\n l'écart entre le max et le min est :%d",max-min);//affichage ecart --> 0.5 pt
68    return 0;
69}
70//*****
71//***** Exercice 3 : 5 pts *****
72//*****
73#include <stdio.h>
```

```

74 int tab[21],milieu,i,n,t;-----> 0.25 pt
75
76 int main(){
77     printf(" Donner la taille de la liste <= 21 :");//
78     scanf("%d",&n);                                     //-----> 0.5 pt
79     milieu=n/2;//-----> 1 pt
80     printf("\n le premier element de la liste ");
81     scanf("%d",&tab[milieu]);//-----> 0.5 pt
82     t=2;
83     for(i=1;i<=milieu;i++){//-----> 0.5 pt
84         printf("\n Donner l'element n°%d :",t++);
85         scanf("%d",&tab[milieu+i]);//-----> 1 pt
86         printf("\n Donner l'element n°%d :",t++);
87         scanf("%d",&tab[milieu-i]);//-----> 1 pt
88     }
89
90     printf("\n le vecteur est :\n");//-----> 0.25 pt
91     for(i=0;i<n;i++){//-----> 0.25 pt
92         printf("%3d",tab[i]);//-----> 0.25 pt
93     }
94
95     return 0;
96 }
97 //***** Exercice 4 : 6 pts *****
98 //***** Exercice 4 : 6 pts *****
99 //***** Exercice 4 : 6 pts *****
100 #include <stdio.h>
101 int vect[100],mat[50][50],i,j,k,l,
102 n,nl,nc,decal,tmp,t;
103 int main(){
104     printf(" Donner la taille du tableau <= 100 :");           //Aquisition des inputs
105     scanf("%d",&n);
106     printf("\n Donner les elements du tableau ");
107     for(i=0;i<n;i++){
108         printf("\n Donner l'element n°%d :",i+1);
109         scanf("%d",&vect[i]);
110     }
111     printf("\n Donner le nombre de ligne de la matrice <= 50 :");
112     scanf("%d",&nl);
113     printf("\n le nombre de colonnes de la matrice est donc %d", n/nl);
114     printf("\n Donner les indices ligne et colonne de la case \n "
115             " de debut de remplissage (numérotation depuis 1):");
116     scanf("%d %d",&k,&l);                                         //-----> 1 pt
117
118     nc=n/nl;                                                       //traitements de remplis-
119     decal=(k-1)*nc+l-1;                                         //sage
120     for(i=1;i<=decal;i++){
121         tmp=vect[n-1];
122         for(j=n-1;j>0;j--){
123             vect[j]=vect[j-1];
124         }
125         vect[0]=tmp;
126     }
127     t=0;
128     for(i=0;i<nl;i++)for(j=0;j<nc;j++)mat[i][j]=vect[t++]; //-----> 4 pts
129
130     printf("\n la matrice est :\n");                                //Affichage matrice
131     for(i=0;i<nl;i++){
132         for(j=0;j<nc;j++){
133             printf("%3d",mat[i][j]);
134         }
135         printf("\n");
136     }
137
138     return 0;
139 }
```

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
UNIVERSITE M'HAMED BOUGARA DE BOUMERDES
FACULTE DES SCIENCES, DEPARTEMENT D'INFORMATIQUE

ETLD, Durée : 1h 30'
Bareme ≈ Cours (10=3+3+4)+TD (10=5+5)

Module : Initiation à l'algorithme (informatique1)
Filière: MI- S1-2016/2017, Date : 03/01/2017

Vous pouvez utiliser l'opérateur mod en algorithmique (% en C). La calculatrice est interdite.

PARTIE COURS (10 pts)

Exercice 1 :

1. Commenter par vrai ou faux, en justifiant, l'assertion : L'exécution du code C suivant :

```
#include <stdio.h>
#include <string.h>
int main() {
    char ch1[7], ch2[5] = "Bon", ch3[5] = "bon";
    ch1[0] = '\0'; strcpy(ch1, ch2); strcat(ch1, ch3); ch1[5] = '\0';
    printf("%s", ch1);
    return 0;
}
```

affichera Bonbon

2. Soit le code algorithmique suivant :

```
Algorithme manipMat ;
    <Partie déclaration>
Debut
    <Lecture de la matrice>
    somLigne ← 0 ;
    Pour i ← 1 à N faire
        Pour j ← 1 à M faire
            somLigne ← somLigne + mat(i,j) ;
            fpour ;
            écrire(somLigne) ;
        fpour ;
Fin.
```

- a) Proposer une partie déclaration pour cet algorithme.
- b) Proposer un code pour lire la matrice dans <Lecture de la matrice>.
- c) Que fait cet algorithme (en une phrase) ?

3. Soit le code suivant écrit en langage algorithmique.

```
Action simple(ES/ C,B,A:entier; E/choix:entier) :
    var tmp:entier;
debut
    si choix=0 alors tmp←C;C←B;B←tmp;
    sinon
        si choix=1 alors C←B; B←C; choix←0;
        sinon C←B+A; B←0; A←0;
        fsi;
    fsi;
fin;
Algorithme tresSimple;
    var A,B,C,D : entier;
debut
    lire(A,B,C,D) ;
    simple(A,B,C,D) ;
    écrire('A=' ,A, ' B=' ,B, ' C=' ,C, ' D=' ,D) ;
fin.
```

On suppose que l'utilisateur entre les valeurs 12 pour A, 34 pour B, 56 pour C et une autre valeur pour D. Commenter par vrai ou faux, en justifiant, les assertions suivantes :

- a) si la valeur entrée de D est 0, `tresSimple` affichera **A=12 B=56 C=34 D=0**
- b) si la valeur entrée de D est 1, `tresSimple` affichera **A=34 B=34 C=56 D=1**
- c) si la valeur entrée de D est -1, `tresSimple` n'affichera rien car ce choix n'est pas pris en considération par l'action paramétrée `simple`.

PARTIE TD (10 pts)

Exercice 2 (5 pts)

Un tableau est dit paritairement trié ssi les éléments ayant des indices de même parité sont triés.

Exemple :

Le tableau

5	4	7	6	9	12	10	23
---	---	---	---	---	----	----	----

 est paritairement trié car $4 \leq 6 \leq 12 \leq 23$ et $5 \leq 7 \leq 9 \leq 10$;

mais

1	2	4	5	8	3	13	10
---	---	---	----------	---	----------	----	----

 ne l'est pas car $5 > 3$.

Écrire un algorithme ou un programme en C qui permet de trier paritairement un tableau d'entiers (taille ≤ 20) sans utiliser un tableau supplémentaire.

Exercice 3 (5 pts)

Étant donnée une liste ordonnée de chiffres décimaux, on veut déterminer tous les nombres impairs (ayant au plus 3 chiffres) pouvant être écrits à partir de cette liste en utilisant des chiffres consécutifs de cette liste respectant l'ordre de celle-ci.

Exemple :

Par exemple, pour la liste 5,1,2,9,3 les nombres à générer sont : 3, 93, 293, 9, 29, 129, 1 et 51 uniquement.

Écrire un algorithme ou un programme en C qui permet de :

1. Lire la liste de chiffres (de taille ≤ 20) dans un tableau d'entiers (Pour l'exemple, l'utilisateur introduit les chiffres 5 puis 1 puis 2, ... etc., jusqu'à 3).
2. Calculer et afficher les nombres impairs générés par la liste.

Il est interdit de se prêter ni effaceur ni quoique ce soit entre étudiants.

L'utilisation du téléphone portable est strictement **interdite** et sera assimilée

a une tentative de **fraude**.

Partie Cours: (3+2,5+4,5)

Ex 1:

1

- 1) Faux, strcpy(ch1, ch2) met "Bon" ds ch1 qui est de taille 7 caractères max. Puis strcat(ch1, ch3) concatène "Bon" avec "bon" donnant "Bonbon", i.e. les six caractères plus le '\0' ds ch1. Puis $ch1[5] = '\0'$ modifie le tab comme suit:

$\begin{array}{ccccccc} 'B' & 'o' & 'n' & ' ' & 'b' & 'o' & '\0' \\ \circ & 1 & 2 & 3 & 4 & 5 & 6 \end{array}$ 0,5

donc le prg affichera Bonbo uniquement.

0,5

2) a: Declaration:

var somLigne, i, N, M, j : entier;

mat : tableau (10, 10) entier;

0,5

b: lire(N, M); { on précisera ds le message que $N \neq M \leq 10$ }
Pour i $\leftarrow 1 \dots N$ faire

Pour j $\leftarrow 1 \dots M$ faire

1

lire(mat(i, j));

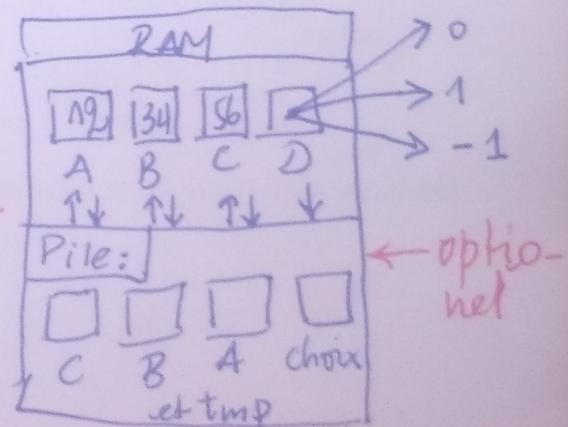
fpour;

fpour;

c: Il calcule les sommes de chaque ligne de façon cumulée. 1

3) Actions paramétrées

a: D=0 donc choix reçoit 0. Le code correspondant permute entre C et B locales donc après $C=12$ et $B=34$, les variables deviennent $C=34$ et $B=12$. Le mode étant ES, on copie c ds A et B ds B. Donc on aura ds l'algo principal $A=34, \dots$



La valeur de A étant \neq de 12, la réponse donnée est fausse. ②
bz $D=1$, donc choix reçoit 1, on exécute $C \leftarrow B; B \leftarrow C$. B étant égale à 34, on aura $C=B=34$. Lors du cadre A reçoit la valeur de C donc 34 ; B globale reçoit celle de B locale donc 34 ; C globale celle de A locale donc 56. D étant transmise en mode entrée n'est pas modifiée. Par conséquent la réponse donnée est juste.

cz $D=-1$ envoyé à l'action simple ds choix est pris en charge par le 2ième sinon donc la réponse donnée est fausse. 1,5

Partie TD = (5+5)

Ex 2: (Une solution abrégée est postée ds mon blog algocriif.)
Algo triParit ;
<declaration> 0,5

debut

<lecture du tab> 0,5

Si $(N \bmod 2) = 0$ alors

 sinon indPairMax $\leftarrow N$; indImpairMax $\leftarrow N-1$;
 fni; indPairMax $\leftarrow N-1$; indImpairMax $\leftarrow N$;

i $\leftarrow 1$; tant que $i \leq \text{indImpairMax}-1$ faire

 j $\leftarrow i+2$;

 tant que $j \leq \text{indImpairMax}$ faire

 Si Vect(j) < Vect(i) alors

 tmp $\leftarrow \text{Vect}(j)$; Vect(j) $\leftarrow \text{Vect}(i)$;
 Vect(i) $\leftarrow \text{tmp}$;

 fni;

 fntq; j $\leftarrow j+2$;

 fntq; i $\leftarrow i+2$;

1,5

$i \leftarrow 2;$
 tant que $i \leq \text{ind Paix Max} - 1$ faire
 $j \leftarrow i + 2;$
 tant que $j \leq \text{ind Paix Max}$ faire
 si $\text{Vect}(j) < \text{Vect}(i)$ alors
 $\text{tmp} \leftarrow \text{Vect}(j); \text{Vect}(j) \leftarrow \text{Vect}(i); \text{Vect}(i) \leftarrow \text{tmp};$
 tri;
 $j \leftarrow j + 2;$
 ffq;
 $i \leftarrow i + 2;$
 ffq;
 <affichage du tableau> 0,5

FIN.EX3:

$\#include <\text{stdio.h}>$
 $\text{int main}()$ {
 0,25 }
 {
 int i, N, vect[20];
 printf("nbr de chiffres(<=20):");
 scanf("%d", &N);
 printf("Donner les chiffres:");
 for(i=0; i<N; i++) { scanf("%d", &vect[i]); }
 printf("Les nbr generes sont:");
 for(i=0; i<N; i++){
 if(vect[i] % 2){
 printf("%d ", vect[i]);
 if(i+1 <= N-1) printf("%d ", vect[i]+10*vect[i+1]);
 if(i+2 <= N-1) printf("%d ", vect[i]+10*vect[i+1]+100*vect[i+2]);
 }
 }
 }

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
UNIVERSITE M'HAMED BOUGARA -BOUMERDES
FACULTE DES SCIENCES
DEPARTEMENT D'INFORMATIQUE

Nature de l'examen : Rattrapage
Durée : 1h

Module : Algorithmique
Filière : MI- S1-2014/2015

Exercice 1(12pts)

Soit un tableau monodimensionnel de type entier A de taille N^2 .

Écrire un algorithme/programme permettant d'effectuer les opérations suivantes :

- 1- Lire le tableau A.
- 2- Construire à partir de A le tableau monodimensionnel B de type entier et de taille N^2 tel que chaque élément de B est évalué comme suit :

$$B[i] = \begin{cases} A[i] & \text{si } i=0 \text{ ou } i=N-1 \\ A[i-1]+A[i]+A[i+1] & \text{sinon} \end{cases}$$

- 3- Copier tous les éléments de B dans un tableau bidimensionnel de type entier Mat [N][N]. Pour cela, B sera parcouru de $B[0]$ à $B[N-1]$ et Mat remplie, ligne par ligne, en commençant par la première.

Exemple

Le tableau B :

-5	6	7	-12	2	10	3	-8	-4	45	1	5	2	4	9	4
----	---	---	-----	---	----	---	----	----	----	---	---	---	---	---	---

Le tableau Mat :

	0	1	2	3
0	-5	6	7	-12
1	2	10	3	-8
2	-4	45	1	5
3	2	4	9	4

- 4- Calculer la trace de Mat définie comme suit :

Mat = Maximum (somme des éléments de ligne impaire de la première diagonale principale, somme des éléments de ligne impaire de la deuxième diagonale principale).

Exemple

	0	1	2	3
0	-5	6	7	-12
1	2	10	3	-8
2	-4	45	1	5
3	2	4	9	4

$$\text{Trace(Mat)} = \text{Max}(10+4, 3+2) = \text{Max}(14, 5) = 14.$$

- 5- Afficher le tableau Mat;
- 6- Afficher la trace de Mat.

Exercice 2(8pts)

Soit A[N][N] un tableau bidimensionnel d'entiers.

On suppose que si on parcourt ce tableau de gauche à droite et ligne par ligne, en commençant par la première, on trouve que les éléments sont triés selon un ordre strictement croissant.

Écrire un algorithme/programme C qui permet d'effectuer les opérations suivantes :

- 1- Lire A
- 2- Vérifier si A respecte le tri supposé.
- 3- En respectant ce tri, insérer un entier dans le tableau A, lorsqu'il n'existe pas déjà. Si l'insertion est réalisée, afficher l'élément sortant du tableau A après le décalage.

Exemple :

1	2	5
7	14	18
27	30	31

Soit le tableau A :

1	2	4
5	7	14
18	27	30

L'insertion de la valeur 4 dans A le transforme en :

L'entier sortant est : 31

Corrigé Rattrapage S1-2014-2015-Initiation à l'algorithmique

Barème

Exercice 1 : (12 pts)

Question 1-	1pt
Question 2-	3 pts
Question 3-	3 pts
Question 4-	3.5 pts
Question 5-	1 pts
Question 6-	0.5 pts

Exercice 2 : (8pts)

Question1	1pt
Question 2	2.5 pts
Question 3	4.5 pts

Exercice 1

```
#include<stdio.h>
#include<stdlib.h>

void main(void)
{
    int A[100],B[100];
    int Mat[100][100];
    int N,M,i,j,k, Trace1,Trace2,Trace;
```

// Q1 : Lecture du tableau A.

```
printf("Entrer la dimension du tableau A \n");
scanf("%d",&N);

for (i=0;i<N*N;i++)
{printf("A[%d]=",i);
 scanf ("%d",&A[i]);
}
```

// Q2 : Construction de B

```
B[0]=A[0];
B[N*N-1]=A[N*N-1];
for (i=1;i<N*N-1;i++)
{    B[i]=A[i-1]+A[i]+A[i+1];
```

```

    }
// Affichage du tableau B
printf("Le tableau B :\n");

for(i=0;i<N*N;i++)
printf("%d ",B[i]);

printf("\n");

//Q3 : Construction de Mat
for (i=0;i<N;i++)
for (j=0;j<N;j++)
Mat[i][j]=B[i*N+j];

// Q4 : Trace de Mat

Trace1=0; Trace2=0;i=1;
while (i<N)
{
    Trace1= Trace1+Mat[i][i];
    i=i+2;
}

i=1; j=N-2;
while (i<N)
{
    Trace2= Trace2+Mat[i][j];
    i=i+2;j=j-2;
}

if (Trace1>Trace2)
Trace=Trace1;
else
Trace=Trace2;

// Q5: Affichage du tableau Mat
printf("Le tableau Mat :\n");

for(i=0;i<N;i++)
{
    for(j=0;j<N;j++)
printf("%d ",Mat[i][j]);
printf("\n");
}

// Q6 : Affichage de la Trace

printf("Trace de Mat = %d \n", Trace);

}

```

Exercice 2

```
#include <stdio.h>
#define N 3
int main()
{
```

```
    int A[N][N],i,j,element,decal=0,trie=1,temp,valdecal,sortant;
```

// Question 1 lecture de A

```
    printf(" entrer les elements du tableau\n");
    for(i=0 ; i<N ; i++)
        for(j=0 ; j<N ; j++)
            scanf("%d",&A[i][j]);
```

// Question 2 verifier si A respecte le tri supposé

```
    for(i=0 ; i<N&&trie ; i++)
        for(j=0 ; j<N&&trie ; j++)
            if(j!=N-1)
                if(A[i][j]>A[i][j+1])
                    trie=0;
            else
                if(i!=N-1)
                    if(A[i][j]>A[i+1][0])
                        trie=0;
```

```
    printf("trie=%d\n",trie);
```

```
    if(!trie)
        printf(" tableau non trié\n");
```

// Question 3 insertion d'une valeur dans A si c'est possible

```
if (trie)
{
    // lecture de la valeur à insérer
    printf(" la valeur à insérer : ");
    scanf("%d",&element);
```

```
    for(i=0 ; i<N ; i++)
        for(j=0 ; j<N ; j++)
            if(decal)
            {
                temp=A[i][j];
                A[i][j]=valdecal;
                valdecal=temp;
```

```
}
```

```

else
    if( element < A[i][j] )
    {
        decal=1;
        printf("ici");
        sortant=A[N-1][N-1];
        valdecal=A[i][j];
        A[i][j]=element;
    }

if(decal)
{
    for(i=0 ; i<N ; i++)
        for(j=0 ; j<N ; j++)
            printf("%d",A[i][j]);

    printf("sortant = %d",sortant);
}
else
    printf("non inseré");
}
}

```

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
UNIVERSITE M'HAMED BOUGARA –BOUMERDES
FACULTE DES SCIENCES
DEPARTEMENT D'INFORMATIQUE

Nature de l'examen : Rattrapage-Dettes
Durée : 1h

Module : Algorithmique
Filière : MI- S1-2013/2014

Exercice 1 (10 pts) :

Soit **Mat[N][N]** un tableau bidimensionnel entier.

Ecrire un algorithme ou un programme permettant d'effectuer les opérations suivantes :

1. Initialiser le tableau **Mat**.
2. Afficher le tableau **Mat**.
3. Construire un tableau **C[N]**, où chaque élément **C[i]** contient le plus grand écart (i.e. : valeur absolue de la différence de deux éléments) entre deux éléments de la ligne i de **Mat**.
4. Afficher le tableau **C**.

Exemple :

En entrée : **Mat[4][4]**

5	-1	9	7
2	5	-1	4
2	3	8	6
0	-5	7	9

En sortie : **C[4]**

10
6
6
14

Exercice 2 (10 pts):

Soit **A[N]** un tableau de N entiers strictement positifs.

Ecrire un algorithme ou un programme permettant d'effectuer les opérations suivantes :

1. Initialiser le tableau **A**.
2. Placer dans **A** d'abord les nombres pairs dans les premières cases de **A**, suivis après par les nombres impairs, en respectant l'ordre de saisie des nombres et sans utiliser un deuxième tableau.
3. Afficher **A**.

Exemple :

En entrée : **A:**

5	3	4	8	7	14	11	2
---	---	---	---	---	----	----	---

En sortie : **A**

4	8	14	2	5	3	7	11
---	---	----	---	---	---	---	----

Corrigé + barème du sujet de rattrapage : Algorithmique S1-2013-2014

<u>Exercice1</u>	<u>Exercice2</u>
<ul style="list-style-type: none"> - Déclaration-----0.5 pts - Initialisation-----1.5 pts - Affichage de Mat-----1 pts - Traitement-----6 pts - Affichage de c -----1 pts 	<ul style="list-style-type: none"> - Déclaration-----0.5pts - Initialisation-----1.5 pts - Traitement-----7 pts - Affichage de c -----1 pts

Exercice1

```
#include <stdio.h>
#include <stdlib.h>
# define n 4
int main()
{   /*Declaration*/
    int mat[n][n],c[n];
    int i,j,max, min;

    /* Initialisation de mat par la lecture */
    for(i=0;i<n;i++)
        for(j=0;j<n;j++)
            {printf("\n a[%d][%d]=",i,j);
            scanf("%d",&mat[i][j]);
            }
    /*Affichage de mat*/
    for(i=0;i<n;i++)
        {for(j=0;j<n;j++)
        //printf("\n a[%d][%d]=%d",i,j,mat[i][j]);
        printf("%d\t",mat[i][j]);
        printf("\n");
        }
    /* Traitement */
    for(i=0;i<n;i++) c[i]=0; // initialiser c

    for(i=0;i<n;i++)
    { max = mat[i][0];min=mat[i][0];
        for(j=0;j<n;j++)
            if (mat[i][j]>max) max=mat[i][j];
            else
                if (mat[i][j]< min) min=mat[i][j];

        c[i]=-(max-min);
    }
    /*Affichage de c*/
    for(i=0;i<n;i++) printf("\n c[%d]=%d", i,c[i]);

    return 0;
}
```

Exercice2

```
#include <stdio.h>
#include <stdlib.h>
#define n 8

int main()
{    /* Déclaration*/
    int a[n];
    int i,j,sauv;

    /* Lecture & affichage de a */

    for(i=0;i<n;i++)
        do scanf("\n %d",&a[i]);while(a[i]<=0);

    for(i=0;i<n;i++)printf("\n a[%d]=%d", i,a[i]); // en plus

    /* Traitement */

    for(i=0;i<n;i++)
        if ((a[i]%2)==1)
            { j=i+1;
              while ((j<n)&&(a[j]%2)==1)j++;
              if (j<n)
                  {sauv=a[j];
                   while (j>i) {a[j]=a[j-1];j--;}
                   a[i]=sauv;
                  }
            }
    /* Affichage de a*/
    printf("\t\t tableau resultant \n\n");
    for(i=0;i<n;i++) printf("\n a[%d]=%d", i,a[i]);

    return 0;
}
```

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
UNIVERSITE M'Hamed BOUGARA –BOUMERDES
FACULTE DES SCIENCES
DEPARTEMENT D'INFORMATIQUE

Nature de l'examen : ETLD
Durée : 1h30

Module : Algorithmique
Filière: MI- S1-2013/2014

Exercice 1 (5pts)

On considère le langage C. Commenter par vrai ou faux, en justifiant, les assertions suivantes :

1. Il n'y a pas de différence entre `i++` et `++i`.
2. Après exécution de l'instruction `scanf("%d", x)`, la variable entière `x` n'est pas modifiée.
3. Dans l'instruction `if(x=1) {x++}`, la condition est évaluée à vrai.
4. `if(x>0) {y=1; z=3}` est équivalente à `if(x>0) y=1; z=3`.
5. Une fonction de type `void` retourne un entier.

Exercice 2 (6 pts)

On considère un tableau d'entiers positifs Vect de taille n . Écrire un algorithme ou un programme C qui détermine le deuxième plus grand écart entre deux éléments consécutifs de ce tableau.

Exemple : Soit `Vect=(1,8,3,10,9,5,8,13,7,4)` un tableau de 10 entiers. Le deuxième plus grand écart est celui entre 13 et 7 et il vaut 6.

NB : L'écart entre deux nombres est la valeur absolue de leur soustraction. i.e. l'écart entre 4 et 5 est égal à l'écart entre 5 et 4 et cet écart vaut 1.

Exercice 3 (9 pts)

On considère une matrice carrée $n \times n$ de nombres entiers. Écrire deux algorithmes ou deux programmes C qui calculent la somme des nombres de cette matrice dont la position définit une structure de damier incluant le premier élément en utilisant :

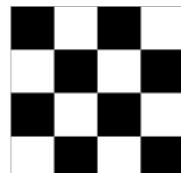
1. un parcours ligne par ligne pour le premier algorithme;
2. un parcours en diagonal pour le deuxième algorithme.

Exemple : Soit la matrice Mat et le damier associé.

Mat=

5	30	2	15
1	12	4	7
16	2	8	9
14	3	11	18

 et le damier associe est :



. Les nombres à additionner sont

ceux des positions noires de ce damier. C'est à dire, pour un parcours ligne par ligne il faut additionner les éléments de Mat dans cet ordre $5+2+12+7+16+8+3+18$; tandis qu'un parcours en diagonale donne $2+7+5+12+8+18+16+3$. Le résultat des deux sommes étant 71.

NB : dans cet exemple $n=4$. La solution demandée doit traiter le cas général.

```
*****  
//Exercice 1*****  
*****  
1. Faux. i++ utilise la valeur de i dans l'expression avant incrementation  
par contre ++i utilise la valeur de i apres incrementation.  
2. Vrai. le deuxieme argument de scanf doit etre transmis par variable  
il faux donc ajouter &.  
3. Vrai. apres affectation x vaut 1 qui est evalue a vrai car non nul.  
4. Faux. z=3 est conditionnelle dans la premiere, imperative dans la  
deuxieme.  
5. Faux. une fonction de type void ne retourne rien d'util.  
  
*****  
//Exercice 2*****  
*****  
#include <stdio.h>  
#include <stdlib.h>  
#include <math.h>  
#include <time.h>  
  
int i,max1,max2,iMax1,iMax2,n,max2Exist;  
int main()  
{  
    printf("\n Donner la taille du tableau :");  
    scanf("%d",&n);  
  
    int tab[n];  
    srand(time(NULL));  
    for(i=0;i<n;i++)tab[i]=rand()%10;  
  
    max1=fabs(tab[0]-tab[1]);  
    iMax1=0; //iMax1 et iMax2 en plus  
    max2Exist=0;  
  
    for(i=1;i<n-1;i++)  
        if(fabs(tab[i]-tab[i+1])>max1){  
            max2Exist=1;  
            max2=max1;  
            iMax2=iMax1;  
            max1=fabs(tab[i]-tab[i+1]);  
            iMax1=i;  
        }  
  
    printf("\n Tab:");  
    for(i=0;i<n;i++)printf("\t%d",tab[i]);  
    if(max2Exist){  
        printf("\n Deuxieme + grand ecart:%d",max2);  
        //printf("\n iMax2:%d",iMax2);  
    }  
    else{  
        printf("\n Deuxieme + grand ecart inexistant");  
    }  
  
    return 0;  
}  
*****  
//Exercice 3*****  
*****  
#include <stdio.h>  
#include <stdlib.h>  
#include <time.h>  
  
int i,j,k,n,lastOddIndex,somParDiag,somParLigne,mySwitch;  
  
int main()  
{  
//Info matrice  
    printf("\n Donner la taille de la matrice :");  
    scanf("%d",&n);  
    //ici au lieu de lire j'utilse des val aleatoires  
    int mat[n][n];  
    srand(time(NULL));
```

```
for(i=0;i<n;i++)for(j=0;j<n;j++)mat[i][j]=rand()%10;

//Question 1 ****
//initialisation
mySwitch=1;
somParLigne=0;

//sommation
for(i=0;i<n;i++){
    for(j=0;j<n;j++){
        if(mySwitch){
            somParLigne+=mat[i][j];
            printf("\ni=%d j=%d",i+1,j+1);
        }
        mySwitch=1-mySwitch;
    }
    if(n%2==0)mySwitch=1-mySwitch;
}

//Question 2 ****
//initialisation
//plus grand indice impair (en C il est pair car index from zero)
lastOddIndex=n-1;
if(!(n%2))lastOddIndex--;
//somme des elem en diagonal du triangle superieur,
//la diagonal principale incluse.
somParDiag=0;
for(k=lastOddIndex;k>=0;k-=2)
    for(i=0,j=k;i<n;i++,j++)somParDiag+=mat[i][j];

//on ajoute ceux en diagonal du triangle inferieur
for(k=2;k<=lastOddIndex;k+=2)
    for(i=k,j=0;i<n;i++,j++)somParDiag+=mat[i][j];

//Affichage
//si les valeurs sont differentes vous me le faites savoir
printf("\n La somme du damier avec parcours ligne/ligne:%d",somParLigne);
printf("\n La somme du damier avec parcours en diag:%d",somParDiag);
return 0;
}
```

Barème

Exo 1 (5pts)

1,....., 5 ----- 5x (réponse : 0.5, Justification 0.5)

Exo 2 (6pts)

- Déclaration des variable nécessaires----- 1.5
- Initialisation du tableau----- 1
- Détermination du deuxième plus grand écart ----- **2**
- Détection du cas où toutes les valeurs sont identiques ----- 1
- Affichage des résultats----- 0.5

Exo 3 (9pts)

- Déclaration des variables nécessaire-----1.5 si un algo. , (0.75x 2) deux algo.
- Initialisation du damier ----- 1 si un algo., 2x 0.5 si deux algo
- Calcul de la somme par ligne ----- **2.5**
- Calcul de la somme par diagonale ----- **3.5**
- Affichage des sommes -----0.5 si un algo., 2x 0.25 si deux algo

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
UNIVERSITE M'HAMED BOUGARA -BOUMERDES
FACULTE DES SCIENCES
DEPARTEMENT D'INFORMATIQUE

Nature de l'examen : ETLD
Durée : 1h30

Module : Algorithmique
Filière: MI- S1-2012/2013

Exercice 1 (6pts)

Soient A et B deux tableaux de N entiers, monodimensionnels triés dans un ordre croissant sans répétitions.

Ecrire un algorithme qui lit A et B puis, détermine un tableau contenant les éléments communs à A et à B, dans un ordre croissant sans répétitions.

Exemple :

Pour A=

1	3	5	7	9
---	---	---	---	---

 et B=

1	5	6	8	9
---	---	---	---	---

les éléments en commun entre A et B sont :

1	5	9
---	---	---

Exercice 2 (7pts)

Dans la compagnie d'assurance TADHAMOUN, les clients payent les 10 % des frais de réparation de leurs véhicules accidentés, avec un apport minimal de 1000 DA et maximal de 4000 DA.

Exemple :

Montant des réparations	10 %du montant	Somme payée par l'assuré	Reste à payer par l'assurance
800	80	800	0
1300	130	1000	300
13000	1300	1300	11700
60000	6000	4000	56000

Ecrire un programme en langage c qui lit le montant total des réparations, puis donne la somme à payer par l'assuré et le reste à payer par l'assurance.

Exercice 3 (7pts)

Soient A un tableau monodimensionnel de n entiers et k un nombre inférieur à n.

Ecrire un algorithme permettant d'effectuer les opérations suivantes :

- Lire puis afficher les éléments de A.
- Déterminer à partir de A la séquence de k nombres consécutifs ayant la plus grande somme.
- Afficher cette séquence ainsi que sa somme.
-

Exemple :

Pour A=

2	1	3	2	2	3	1
---	---	---	---	---	---	---

 avec n=7 et k= 3, la séquence ayant la plus

grande somme est 3 2 2 et sa somme est 7.

Corrigé

Exercice 1

```
Algorithme intersection
Const n = 10;
Entier a[n], b[n], c[n], i, j, k;
Debut
    /*Lecture */
    Ecrire("Lecture du tableau A:");
    Pour i=0 à n
    Debut
        Ecrire("a[",i,"]=");
        Lire (a[i]);
    Fin

    Ecrire("Lecture du tableau B:");
    Pour i=0 à n
    Debut
        Ecrire("b[",i,"]=");
        Lire (b[i]);
    Fin

    /*Traitement*/
i=0; j=0; k=0;
Tantque ((i<n) ET (j<n))
    Debut
        Si (a[i]= b[j])
            Debut
                C[k]=a[i];
                i=i+1;
                j=j+1;
                k=k+1 ;
            Fin
            sinon si(a[i]<b[j])
                i=i+1;
            sinon j=j+1;
        Fin

    /*Affichage*/
    Ecrire("les nombres en commun sont:");
    Pour i=0 à k-1
        Ecrire("c[",i,"]=",c[i])
    Fin.
```

Exercice 2

```
#include <stdio.h>
void main()
{
double montant,dixp,client;

/*Lecture */
printf("Entrer le Montant des frais de réparation:");
scanf("%lf",&montant);

/*Traitement*/
dixp=montant*0.1;

if (montant<1000) client=montant;
else if (dixp<1000) client=1000;
else if (dixp<4000) client=dixp;
else client=4000;

/* affichage */
printf("Somme à payer par l'assuré:%f",client);
printf("Somme à payer par l'assurance:%f",montant-client);
}
```

Exercice 3

```
Algorithme grandesequence
Const n=10;
Entier a[n],i,j,k,s,max,pos;

Debut
    /*Lecture et affichage du tableau */
    Ecrire("Entrer la longueur de la séquence");
    Lire(k);

    Ecrire("Entrer les éléments du tableau A:");
    Pour i=0 à n-1
        Debut
            Ecrire("a[",i,"]=");
            Lire(a[i]);
        Fin

    Ecrire("Affichage du tableau A:");
    Pour i=0 à n-1
        Ecrire("a[",i,"]=",a[i]);

    /*Traitement*/
    s=0;
    Pour i=0 à k-1
        s=s+a[i];

    max= s; pos=0;
    Pour i=1 à n-k
        Debut
            s=0;
            Pour j=i à i+k-1
                s=s+a[j];

            If (s>max)
                Debut
                    max=s; pos=i;
                Fin
            Fin

    /* affichage de la séquence et de sa somme */
    Ecrire("La séquence ayant la plus grande somme:");
    Pour i=pos à pos+k-1
        Ecrire(a[i]);
    Ecrire("La somme de cette séquence: ",max );

Fin.
```