TDs Language C:

TD 1:

EXERCICE 1:

1. a = x + 5
2. a = (x = y) + 2
3. a = x == y
4. a < b * c < d
5. i ++ * (n +p)

EXERCICE 2:

1* short int
2* int
3* int
4* int

EXERCICE 3:

1 * long
2 * float
3 * char
4 * float

EXERCICE 4 :

q = 1 ;
q = 0 ;
q = 1 ;
x = 1 ;
x = 1.80 ;
x = 1,90 ;
x = 1 ;
q = 25 ;
q = 45 ;

EXERCICE 5 :

A : i = 1, n = 0.
B : i = 11,n = 11.

C : i = 21,j = 6,n = 120.
D : i = 18, n = 18.
E : i = 12,j = 4,n = 12.

EXERCICE 6 :

A : n = 10, p = 10,q = 10,r = 1.
B : n = 15,p = 10,q = 5.
C : n =15,p = 11,q = 10.
D : n = 16, p = 11,q = 15.

EXECICE 7 :

A : n = 6, p = 2,q = 1.
B : n = 5,p = 3,q = 1.
C : n = 6,p = 2,q = 0.
D : n = 6, p = 3,q = 1.

TD 2:

EXERCICE 1 :

A : 543 34.567800
B : ^543 ^34.567800
C : 543 34.567800
D : ^^^^34.567 3.457e + 001
E : 543^^ 34.567800
F : ^^^543
G : ^^^^34.56780

EXERCICE 2 :

A : S
B : S
C : la valeur de S en code ASCII sera affichée 2 fois : 83 83

EXERCICE 3 :

A : n = 253 ; p = 45.
B : n = 253 ; p = 4.

EXERCICE 4 :

A : n = 12 ; p = 45.
B : n = 1234 ; p = 56.
C : n = 1234 ; p = 56.
D : n = 1 ; p = 45.
E : n = 4567 ; p = 89.

EXERCICE 5 :

n = 1 ; p = 2.
n = 3 ; p = 4.
n = 1234 ; p = 56.
n = 7890 ; p = 12.
n = 34 ; p = 5.
n = 6 ; p = 7.
n = 8 ; p = 9.
n = 10 ; p = 0.
n = 0 ; p = 12.

TD 3 :

EXERCICE 1 :

VOIR L'ENONCÉ DU TD .

EXERCICE 2 :

A : NUL
PETIT
B : PETIT
C : MOYEN
GRAND
D : GRAND
E : GRAND

EXERCICE 3 :

VOIR L'ENONCÉ DU TD .

EXERCICE 4 :

```
do {
Printf( '' donnez un nombre > 0 '' ) ;
scanf( ''%d '' , &n) ;
}
while( n<=0) ;
```

EXERCICE 5 :

- :

  A :

  ```
  while(i < 4){
  printf( ''donnez un entier '' ) ;
  scanf( ''%d  '',&n ) ;
  som+=n ;
   }
  ```

- :

  B :

  ```
  do{
  printf( ''donnez un entier '' ) ;
  scanf( ''%d  '',&n ) ;
  som+=n ;
   }
  while(i < 4) ;
  ```

- :

  0 est pair
  n=3
  L'instruction '' continue '' va reinsialiser la boucle.
  3 est multiple de 3
  n= 8
  n=9
  9 est multiple de 3
  n= 14
  …
  boucle infinie.

EXERCICE 6 :

```
#include <stdio.h>
#include <math.h>
int  main()
{

 int A, B, C;
 double D;
```

```
printf("Calcul des solutions réelles d'une équation du second \n");
printf("degré de la forme   ax^2 + bx + c = 0 \n\n");
printf("Introduisez les valeurs pour a, b, et c : ");
scanf("%d %d %d", &A, &B, &C);
D = B*B - 4.0*A*C;
/* Distinction des différents cas */
if (A==0 && B==0 && C==0) /* 0x = 0 */
        printf("Tout réel est une solution de cette équation.\n");
else if (A==0 && B==0)  /* Contradiction: c # 0 et c = 0 */
      printf("Cette équation ne possède pas de solutions.\n");
else if (A==0) /* bx + c = 0 */
  {
   printf("La solution de cette équation du premier degré est :\n");
   printf(" x = %.4f\n", (double)C/B);
  }
else if (D<0)  /* b^2-4ac < 0 */
   printf("Cette équation n'a pas de solutions réelles.\n");
else if (D==0) /* b^2-4ac = 0 */
  {
   printf("Cette équation a une seule solution réelle :\n");
   printf(" x =  %.4f\n", (double)-B/(2*A));
  }
else /* b^2-4ac > 0 */
  {
   printf("Les solutions réelles de cette équation sont :\n");
   printf(" x1 = %.4f\n", (-B+sqrt(D))/(2*A));
   printf(" x2 = %.4f\n", (-B-sqrt(D))/(2*A));
  }
 return 0;
}
```

EXERCICE 7 :

```
#include <stdio.h>
#include <math.h>
int  main()
{
int n,i;
float som=0;
   printf("Entrez le nbr de terme de votre serie harmonique: \n");
   scanf("%d",&n);
   for(i=1;i<n+1;i++){
      som+=(float)1/i;
   }
   printf("Le resultat de votre somme est : %f \n",som);
   return 0;
```

```
}
```

EXERCICE 8 :

```
#include <stdio.h>
#include <math.h>
int  main()
{
int i,j,k,n;
   printf("Entrez le nombre de lignes souhaitees \n");
   scanf("%d",&n);
   for(i=0;i<n;i++){
      for(j=n-1;j>i;j--)
         printf(" ");
      for(k=0;k< 2*i+1;k++)
         printf("*");
      printf("\n");
   }
}
```

EXERCICE 9 : Version récursive :

```
#include <stdio.h>
#include <math.h>
int  fibo ( int n ){
   if(n==1 || n==2)
      return 1;
   else
      return (fibo(n-1) + fibo(n-2));
}
int  main()
{
int n;
   printf("Entrez le nombre de termes : \n");
   scanf("%d",&n);
   printf("u%d = %d\n",n, fibo(n));
}
```

Version iterative

```
#include <stdio.h>
#include <math.h>
int  main()
{
int n,u=1,u1=1,u2=1,i;
   printf("Entrez le nombre de termes : \n");
   scanf("%d",&n);
   for(i=2;i<n;i++){
```

```
      if(n==1 || n==2)
         u=1;
      else{
         u= u1 + u2;
         u1=u2;
         u2=u;
      }
   }
   printf("u%d = %d\n",n,u);
}
```

EXERCICE 10 :

```
#include <stdio.h>
#include <math.h>
int  main()
{
int max=0,min=20,imax=0,imin=0,note=0;
   do{
   printf("donnez une note (-1 pour finir) : \n");
   scanf("%d",&note);
   if (note<=0)
      break;
   else{
      if(note<min){
         min=note;
         imin=1;}
      else if(note==min)
           imin++;
      if(note>max){
         max=note;
         imax=1;}
      else if(note==max)
           imax++;
   }
   }
   while (note>=0);
printf("la note maximale : %d attribuee %d fois \n",max,imax);
printf("la note minimale : %d attribuee %d fois \n",min,imin);
}
```

TD 5:

EXERCICE 1 :

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

```
#include <math.h>
int main(){
void f1(void);
void f2(int);
int f3(int);
int n;
printf("combien de fois voulez vous afficher bonjour \n");
scanf("%d",&n);
f1();
f2(n);
n=f3(n);
   return 0;
}
void f1(void){
   printf("good morning * \n");
}
void f2(int n){
int i=0;
   while(i<n){
    printf("good morning **\n");
      i++;
   }
}
int f3(int n){
int i=0;
   while(i<n){
    printf("good morning ***\n");
      i++;
   }
   return 0;
}
```

EXERCICE 2 :

Le programme affichera : 5 3 .

EXERCICE 3 :

```
#include <stdio.h>
#include <math.h>
void call(void){
static long n=0,limit=1; /* static local variable to keep the same value from call to another*/
n++;
if(n>=limit){
   printf("*** call %ld time *** \n",limit);
   limit*=10;
   }
}
void main(){
```

```
long nmax=10000000,n=0;
while(call(),n++,n<nmax);
printf("*** END OF PROGRAM *** \n");
}
```

EXERCICE 4 :

```
#include <stdio.h>
#include <math.h>
int A(int m,int n){
if(m<=0 && n<=0)
   return 0;
else if(m==0)
   return(n+1);
else if(n==0)
   return(A(m-1,1));
else return(A(m-1,A(m,n-1)));
}
int main()
{
int n,m;
scanf("%d %d",&m,&n);
printf(" le resu de la foncction A est: %d \n", A(m,n));
 }
```

EXERCICE 5 :

Le programme affichera :

```
B : Dans fct : n = 10  ; p = 5 ; q = 20 ;
A : Dans main : n = 20 ; p = 5 ; q = 2 ;
C : Dans f : n = 10 ; p = 20 ; q = 2 ;
```

EXERCICE 6 :

```
#include <stdio.h>
#include <math.h>
void calculator(float m,float n,char c){
   switch (c) {
      case '+':
         printf(" %f + %f : %f \n",m,n,m+n);
         break;
      case '-':
         printf(" %f - %f : %f \n",m,n,m-n);
         break;
      case '*':
         printf(" %f * %f : %f \n",m,n,m*n);
         break;
      case '/':
```

```
            printf(" %f / %f : %f \n",m,n,m/n);
            break;
        case '?':
            break;
    default:
            printf(" %f + %f : %f \n",m,n,m+n);
            break;
     }
}
int main(){
float m,n;
char c;
printf(" Enter 2 number : \n");
scanf("%f %f",&m,&n);
do{
getchar();
printf(" Enter the sign of your operation (type ? if you want to quite) : \n");
scanf("%c",&c);
calculator(m, n,c);
}
while(c !='?');
}
```

EXERCICE 7 :

```
#include <stdio.h>
#include <math.h>
char c;
void calculator(float m,float n){
    switch (c) {
        case '+':
            printf(" %f + %f : %f \n",m,n,m+n);
            break;
        case '-':
            printf(" %f - %f : %f \n",m,n,m-n);
            break;
        case '*':
            printf(" %f * %f : %f \n",m,n,m*n);
            break;
        case '/':
            printf(" %f / %f : %f \n",m,n,m/n);
            break;
        case '?':
            break;
        default:
            printf(" %f + %f : %f \n",m,n,m+n);
            break;
    }
}
```

```
int main(){
float m,n;
printf(" Enter 2 number : \n");
scanf("%f %f",&m,&n);
do{
getchar();
printf(" Enter the sign of your operation (type ? if you want to quite) : \n");
scanf("%c",&c);
calculator(m, n);
}
while(c !='?');
}
```

EXERCCICE 8 :

```
#include <stdio.h>
#include <math.h>
void verification(int n){
if(n%2==0)
   printf("il est pair \n");
else
   printf("il est impair \n");
if(n%3==0)
   printf("il est multiple de 3 \n");
if(n%3==0 && n%2==0)
   printf("il est diviseur de 6 \n");
}
int main(){
int nbr;
printf("Enter a number: \n");
scanf("%d",&nbr);
verification(nbr);
}
```

TD 6 :

EXERCICE 1 :

```
#include <stdio.h>
#include <math.h>
void fill(int tab[],int n){
int i=0,sum=0;
while(i<n){
  printf("value of position %d \n",i+1);
  scanf("%d",&tab[i]);
  sum+=tab[i];
   i++;
    }
printf("Sum of this element array is: %d \n",sum);
```

```
}
void display(int tab[],int n){
int i=0;
while(i<n){
   printf("%d ",tab[i]);
   i++;
   }
}
int main(){
int tab[50],n;
printf("what's the dimension of your array? \n");
scanf("%d",&n);
printf("fill the array..\n");
fill(tab,n);
printf("there's your array: \n");
display(tab,n);
}
```

EXERCICE 2 :

```
#include <stdio.h>
#include <math.h>
void fill(int tab[]){
int i=0;
while(i<10){
  printf("value of position %d \n",i+1);
  scanf("%d",&tab[i]);
   i++;
   }
}
void display(int tab[]){
int i=0;
while(i<10){
printf("%d ",tab[i]);
   i++;
}
}
void max_min(int tab[]){
int i=0,min=tab[0],max=tab[0];
while(i<10){
if(max<tab[i])
   max=tab[i];
if(min>tab[i])
   min=tab[i];
i++;
   }
printf("\n Max = %d ; Min = %d \n",max,min);
}
int main(){
```

```
int tab[10];
printf("fill the array..\n");
fill(tab);
printf("there's your array: \n");
display(tab);
max_min(tab);
}
```

EXECICE 3 :

```
#include <stdio.h>
#include <math.h>
void fill(int tab[],int n){
int i=0;
while(i<n){
  printf("value of position %d \n",i+1);
  scanf("%d",&tab[i]);
  i++;
    }
}
void display(int tab[],int n){
int i=0;
while(i<n){
   printf("%d ",tab[i]);
   i++;
   }
}
void eraser(int tab[],int n,int nbr){
int i=0,j;
while(i<n){
   if(tab[i]==nbr){
      for(j=i;j<n;j++)
         tab[j]=tab[j+1];
         n--;
         i-- ;
   }
   i++;
   }
display(tab,n);
}
int main(){
int tab[50],n,nbr;
printf("what's the dimension of your array? \n");
scanf("%d",&n);
printf("fill the array..\n");
fill(tab,n);
printf("there's your array: \n");
display(tab,n);
printf("\nwhat's the value you want to delate \n");
```

```
scanf("%d",&nbr);
eraser(tab,n,nbr);
}
```

EXERCICE 4 :

```
#include <stdio.h>
#include <math.h>
void fill(int tab[],int n){
int i=0;
while(i<n){
  printf("value of position %d \n",i+1);
  scanf("%d",&tab[i]);
  i++;
    }
}
void display(int tab[],int n){
int i=0;
while(i<n){
    printf("%d ",tab[i]);
    i++;
    }
}
void invert(int tab[],int n){
int i=0,j=n-1,inv;
    while(i<j){
    inv=tab[i];
    tab[i]=tab[j];
    tab[j]=inv;
    j--;
    i++;
    }
display(tab,n);
}
int main(){
int tab[50],n;
printf("what's the dimension of your array? \n");
scanf("%d",&n);
printf("fill the array..\n");
fill(tab,n);
printf("there's your array: \n");
display(tab,n);
printf("\nyour array after inversion.. \n");
invert(tab,n);
}
```

EXERCICE 5 :

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

```c
#include <math.h>
int main(){
int a,b,c,d,e,f;
printf("Fill the first vector element value: \n");
scanf("%d %d %d",&a,&b,&c);
printf("Fill the second vector element value: \n");
scanf("%d %d %d",&d,&e,&f);
printf("Dot product of this two vectors is: \n");
printf("%d * %d + %d * %d + %d * %d = %d \n",a,d,b,e,c,f,a*d+b*e+c*f);
}
```

EXECICE 6 :

```c
#include <stdio.h>
#include <math.h>
void fill(int tab[],int n){
int i=0;
while(i<n){
  scanf("%d",&tab[i]);
   i++;
    }
}
void display(int tab[],int n){
int i=0;
while(i<n){
    printf("%d ",tab[i]);
    i++;
    }
}
void sorting(int tab[],int n,int val){
int i=n;
    for(;i>0 && tab[i-1]>val;i--)
        tab[i]=tab[i-1];
    tab[i]=val;
    n++;
    display(tab,n);
}
int main(){
int tab[50],n,val;
printf("what's the dimension of your array? \n");
scanf("%d",&n);
printf("fill the array..\n");
fill(tab,n);
printf("there's your array: \n");
display(tab,n);
printf("\nAdd a new element \n");
scanf("%d",&val);
sorting(tab,n,val);
```

}

TD 7 :

EXERCICE 1 :

	A	B	C	P1	P2
Init.	1	2	3	/	/
P1=&A	1	2	3	&A	/
P2=&C	1	2	3	&A	&C
*P1=(*P2)++	3	2	4	&A	&C
P1=P2	3	2	4	&C	&C
P2=&B	3	2	4	&C	&B
*P1-=*P2	3	2	2	&C	&B
++*P2	3	3	2	&C	&B
P1=*P2	3	3	6	&C	&B
A=++*P2**P1	24	4	6	&C	&B
P1=&A	24	4	6	&A	&B
*P2=*P1/=*P2	6	6	6	&A	&B

```
main()
   {
      int A = 1;
      int B = 2;
      int C = 3;
      int *P1, *P2;
      P1=&A;
      P2=&C;
      *P1=(*P2)++;
      P1=P2;
      P2=&B;
      *P1-=*P2;
      ++*P2;
      *P1*=*P2;
      A=++*P2**P1;
      P1=&A;
      *P2=*P1/=*P2;
      return 0;
   }
```

EXERCICE 2 :

Écrire Soit P un pointeur qui 'pointe' sur un tableau A:

```
   int A[] = {12, 23, 34, 45, 56, 67, 78, 89, 90};
   int *P;
   P = A;
```

```
Quelles valeurs ou adresses fournissent ces expressions:
       a)     *P+2        14
       b)     *(P+2)      34
       c)     &P+1        Adresse inconnue.
       d)     &A[4]-3     &A[1]
       e)     A+3         A[3]
       f)     &A[7]-P     7
       g)     P+(*P-10)    &A[2]
       h)     *(P+*(P+8)-A[7])     23
```

EXERCICE 3 :

1 * : Remplissage du tableau Ad par les valeurs du tableau t.
2 * : 10 20 30 40.
3 * : 30 21.

EXERCICE 4 :