Note :I am not responsible for any mistakes in this document

concerning

1. If you can't understand my hand writing please verify the maths record pdf given by college

* The maths record programs,
Maths record outputs pdf,
Are given in maths file
Please check it

-SETHU

1. Addition, subtraction of Matrices

AIM:-

To perform addition and subtraction of two matrices using c.

ALGORITHM:-

Step 1: Start
Step 2: Read the values for m, n.
Step 3: Read matrix values of a[i][j] and b[i][j]
Step 4: Initialize i=1
Step 5: If (i<=m) then go to step 6
else goto step 10.
Step 6: Initialize j=1
Step 7: If j<=n then goto step 8
else
Set i= i+1 and goto step 5.
step 8: set c[i][j] = a[i][j] + b[i][j]
d[i][j] = a[i][j] - b[i][j]
j= j+1 goto step 7.
step 9: Display the value of c[i][j]
step 10: Display the value of d[i][j]
Step 11: Stop.

Program:-

```
#include <stdio.h>
#include <conio.h>
Void main ()
{
int a[5][5], b[5][5], c[5][5], d[5][5], i, j, m, n;
clrscr();
Printf ("Enter Size of the matrices \n");
scanf ("%d %d", &m, &n);
printf ("Enter matrix A of order %d x %d \n", m, n);
for (i=0; i<m; i++)
    for (j=0; j<n; j++)
        scanf ("%d", &a[i][j]);
printf ("Enter matrix B of order %d x %d \n", m, n);
for (i=0; i<m; i++)
    for (j=0; j<n; j++)
        scanf ("%d", &b[i][j]);
for (i=1; i<=m; i++)
for (j=1; j<=n; j++)
{
c[i][j] = a[i][j] + b[i][j];
d[i][j] = a[i][j] - b[i][j];
}
printf ("A+B= \n");
for (i=1; i<=m; i++)
{
for (j=1; j<=n; j++)
```

```
        printf("%d\t", c[i][j]);
    printf("\n");
}
printf("A-B=\n");
    for (i=1; i<=m; i++)
    {
        for (j=1; j<=n; j++)
            printf("%d\t", d[i][j]);
    printf("\n");
    }
getch();
}
```

Output:-

```
Enter the size of the matrices (mxn):
3 3
Enter Matrix A of order 3X3:
-1 -3 2
2 1 -3
4 -1 3
Enter matrix B of order 3x3:
2 -2 -4
-1 3 4
1 -2 3
A+B =
3 -5 -2
1 4 1
5 -3 6
A-B =
-1 -1 6
3 -2 -7
3 1 0
```

2. Multiplication of Matrices

Aim:-

To perform multiplication of two matrices using C

Algorithm:-

Step1: start
Step2: read size of matrices m1, n1, m2, n2
Step3: if (n1 != m2)
print "matrix multiplication not possible"
and goto step 14.
else
goto step 4
Step4: read matrix value a[i][j] and b[i][j]
Step5: initialize i=1
Step6: if i<=m1 goto step 7 else goto step
Step7: initialize j=1
Step8: if i<=n2 goto step 9
else
print new line, set i=i+1 and goto step 6
Step9: c[i][j]=0
Step10: initialize k=1
Step11: if k<=n1
go to step 12
else
print c[i][j]
set j=j+1 and goto step 8
Step12: c[i][j] = c[i][j] + a[i][k] * b[k][j]
Step13: Set k=k+1 and goto setep 11.
Step14: Stop.

Program :-

```
#include <stdio.h>
#include <conio.h>
void main()
{
int a[10][10], b[10][10], c[10][10], m1, n1, m2, n2, i, j,
    k;
clrscr();
printf("Enter size of matrix A \n");
scanf("%d %d", &m1, &n1);
printf("Enter size of matrix B \n");
scanf("%d %d", &m2, &n2);
if(n1 != m2)
printf(" matrix multiplication not possible.");
else
{
printf("Enter matrix A of order %d x %d \n", m1, n1);
for(i=1; i<=m1; i++)
    for(j=1; j<=n1; j++)
        scanf("%d", &a[i][j]);
printf("Enter matrix B of order %d x %d \n", m2, n2);
for(i=1; i<=m2; i++)
    for(j=1; j<=n2; j++)
    scanf("%d", &b[i][j]);
printf("The product matrix AB is \n");
for(i=1; i<=m1; i++)
{
```

```
for (j=1; j<=n2; j++)
 {
  c[i][j]=0;
  for (k=1; k<=m1; k++)
    c[i][j]=c[i][j]+a[i][k]*b[k][j];
printf("%d\t",c[i][j]);
  }
  printf("\n");
 }
}
getch();
}
```

Output:

```
Enter Size of matrix A
3 3
Enter Size of matrix B
3 3
Enter matrix A of order 3x3
3 -3 4
2 -3 4
0 -1 1
Enter Matrix B of order 3x3
3 1 2
2 0 5
1 2 0
The product matrix AB is
7 11 -9
4 10 -11
-1 2 -5
```

3. Determinant of a matrix and inverse of a matrix

AIM:
to find determinant and inverse of a 3x3 matrix using c

Algorithm:
Step 1: Start
Step 2: Set det=0
Step 3: read values of a 3x3 matrix for a[i][j]
Step 4: initialize i=0
Step 5: if i<3 goto step 6
else
goto step 9
Step 6: initialize j=0
Step 7: if j<3
goto step 8
else
set i=i+1 and goto step 5
Step 8: det=det+(a[0][i]*(a[1][(i+1)%3]*a[2][(i+2)%3]-a[1][(i+2)%3]*a[2][(i+1)%3]))
set j=j+1 and goto step 7
Step 9: print det value
Step 10: if det=0
print "Inverse does not exists" and goto step 14
else
goto step 11
Step 11: Calculate cofactors matrix values
cof[i][j]=((a[(i+1)%3][(j+1)%3]*

a[(i+2)%3][(j+2)%3])
-(a[(i+1)%3][(j+2)%3]*a[(i+2)%3][(j+1)%3]))

step 12: calculate adjoint matrix values
adj[i][j] = cof[j][i]

step 13: print values of a[i][j]/det

step 14: stop

Program :-

```
# include <stdio.h>
# include <conio.h>
  void main ()
  {
  int a[3][3], i, j;
  float cof[3][3], adj[3][3], det=0, t;
  clrscr();
  printf ("Enter a 3x3 matrix: \n");
    for (i=0; i<3; i++)
      for(j=0; j<3; j++)
        scanf ("%d", &a[i][j]);
  for (i=0; i<3; i++)
    det = det + (a[0][i] * (a[1][(i+1)%3] * a[2][(i+2)
          %3] - a[1][(i+2)%3] * a[2][(i+1)%3]));
printf ("\n Determinant of given
        matrix = %f", det);
    if (det==0)
printf("\n inverse doesn't exists for given matrix");
    else
    {
    printf("\n Inverse of matrix is :\n\n");
      for (i=0; i<3; i++)
        for (j=0; j<3; j++)
        cof[i][j] = ((a[(i+1)%3][(j+1)%3]
*a[(i+2)%3][(j+2)%3]) -
(a[(i+1)%3][(j+2)%3] * a[(i+2)%3][(j+1)%3]));
```

```
for(i=0;i<3;i++)
  {
   for(j=0;j<3;j++)
    {
     adj[i][j]=cof[j][i];
     t=adj[i][j]/det;
     printf("%0.2f\t",t);
  printf("\n");
    }
  }
getch();
 }
```

Output:

Enter a 3x3 matrix:

1	2	-2
-1	3	0
0	-2	1

Determinant of given matrix = 1.0

Inverse of matrix is:

3.0	2.0	6.0
1.0	1.0	2.0
2.0	2.0	5.0

4. Singular and non-singular matrixes

AIM:

To check whether a given 3x3 matrix is singular or non-singular matrix using C.

Algorithm:-

step1 : start
step2 : set det=0
step3 : read values of a 3x3 matrix for a[i][j]
step4 : initialize i=0
step5 : if i<3
goto step 6
else
goto step 9
step 6 : initialize. j=0
step7 : if j<3
goto step 8
else
set i=i+1 and goto step 5
step 8 : det = det + (a[0][i] * (a[1][(i+1)%3] * a[2][(i+2)%3] - a[1][(i+2)%3] * a[2][(i+1)%3]))
set j=j+1 and goto step 7
Step 9 : print det value
step10 : if det=0
print "given matrix is singular"
else
print "given matrix is non-singular"
Step 11 : stop.

Program:-

```
#include <stdio.h>
#include <conio.h>
void main()
{
int a[3][3], i, j, det=0;
clrscr();
printf("Enter a 3x3 matrix :\n");
for(i=0; i<3; i++)
    for(j=0; j<3; j++)
        scanf("%d", &a[i][j]);
for(i=0; i<3; i++)
det=det+(a[0][i]*(a[1][(i+1)%3]*a[2][(i+2)%3]
-a[1][(i+2)%3]*a[2][(i+1)%3]));
printf("Determinant of given matrix = %d", det);
if(det==0)
printf("\n given matrix is singular");
else
printf("\n given matrix is non-singular");
getch();
}
```

Output:

Enter a 3X3 matrix:
1 2 1
3 2 2
1 1 2
Determinant of given matrix = -5
Given matrix is non-singular

5. Matrix inversion method

AIM:- To solve a given system of linear equations with 3 unknows by matrix inversion method using C.

Algorithm:-

Step 1: start
Step 2: set det=0
Step 3: read value of a co-efficient matrix a[i][j] and constant matrix b[i][j]
Step 4: initialize i=0
Step 5: if i<3
go to step 6
else
go to step 9
Step 6: initialize j=0
Step 7: if j<3 goto step 8
else
set i=i+1 and goto step 5
Step 8: det=det+(a[0][i]*(a[1][(i+1)%3]*a[2][(i+2)%3]
-a[1][(i+2)%3]*a[2][(i+1)%3]))
Set j=j+1 and goto step 7
Step 9: if det=0
print "solution does not exists"
and goto step 4
else
goto step 11
Step 11: Calculate cofactors matrix values.
cof[i][j]=((a[(i+1)%3][(j+1)%3]*

a[(i+2)%3][(j+2)%3])-
(a[(i+1)%3][(j+2)%3]*a[(i+2)%3][(j+1)%3]))

step 12: calculate adjoint matrix value
adj[i][j]=cof[j][i]

step 13: calculate inverse matrix inv[i][j]=a[i][j]/det

step 16: initialize. j=1

step 17: if j<1
goto step 18
else
print new line, set i=i+1 and goto step 15

step 18: t[i][j]=0

step 19: initialize k=0

step 20: if k<3
goto step 21
else
print t[i][j]
Set j=j+1 and goto step 17

step 21: t[i][j]=t[i][j]+inv[i][k]*b[k][j]

Step 22: Set k=k+1 and goto step 20

step 23: stop

Program :-

```
#include <stdio.h>
#include <conio.h>
void main ()
{
int a[3][3], b[3][1], i, j, k;
float cof[3][3], inv[3][3], t[3][1], det=0;
clrscr();
printf("Enter coefficient matrix: \n");
for (i=0; i<3; i++)
    for (j=0; j<3; j++)
        scanf("%d", &a[i][j]);
printf("enter constant matrix: \n");
for(i=0; i<3; i++)
    for (j=0; j<1; j++)
        scanf("%d", &b[i][j]);
for (i=0; i<3; i++)
det=det+(a[0][i]*(a[1][(i+1)%3]*a[2][(i+2)%3]-
a[1][(i+2)%3]*a[2][(i+1)%3]));
if (det==0)
 printf("solution does not exist");
else
{
printf("\n The values of x, y, z are: \n");
   for(i=0; i<3; i++)
        for(j=0; j<3; j++)
 cof[i][j]=((a[(i+1)%3][(j+1)%3]*
```

```
a[(i+2)%3][(j+2)%3])-(a[(i+1)%3][(j+2)%3]*
a[(i+2)%3][(j+1)%3]));
for(i=0;i<3;i++)
   for(j=0;j<3;j++)
     inv[i][j]=cof[j][i]/det;
 for(i=0;i<3;i++)
    for(j=0;j<1;j++)
   {
    t[i][j]=0;
    for(k=0;k<3;k++)
t[i][j]=t[i][j]+inv[i][k]*b[k][j];
printf("%.2f",t[i][j]);
 printf("\n");
   }
}
getch();
}
```

output:-

```
Enter coefficient matrix:
3   1   2
2   -3   -1
1   2   1
Enter constant matrix:
3
-3
4
The values of x, y, z are:
1.0
2.0
-1.0
```

6. Cramer's Rule

AIM:-

To solve a given system of linear equations with 3 unknowns by matrix cramer's method using C.

Algorithm:

Step 1: Start
Step 2: define a function det() to calculate determinant
Step 3: read values of a co-efficient matrix a[i][j] and set t1[i][j] = t2[i][j] = t3[i][j] = a[i][j]
Step 4: read values of constant matrix b[i][j]
Step 5: Calculate d = det(a)
Step 6: if d=0
print "solution does not exist"
else
goto step 7
Step 7: replace column 1 values of t1[i][j] with b[i][j] then d1 = det(t1)
Step 8: replace column 2 values of t2[i][j] with b[i][j] then d2 = det(t2)
Step 9: replace column 3 values of t3[i][j] with b[i][j] then d2 = det(t3)
Step 10: x = d1/d, y = d2/d, z = d3/d
Step 11: print the values of x, y, z
Step 12: Stop.

Program :-

```
#include <stdio.h>
#include <conio.h>
 float det (int a[3][3])
 {
  float d=0.0;
  for (int i=0; i<3 ; i++)
   d=d+(a[0][i]*(a[1][(i+1)%3]*a[2][(i+2)%3]-
   a[1][(i+2)%3]*a[2][(i+1)%3]));
return d;
 }
void main()
{
int a[3][3]., b[3][1], t1[3][3], t2[3][3], t3[3][3],
     i, j;
float d, d1, d2, d3;
 clrscr();
printf("Enter coefficients of variables: \n");
 for (i=0; i<3; i++)
     for (j=0; j<3; j++)
      {
      scanf("%d", &a[i][j]);
      t1[i][j]=t2[i][j]=t3[i][j]=a[i][j];
      }
printf("Enter constants: \n");
 for (i=0; i<3; i++)
     for (j=0; j<1; j++)
```

```
    scanf("%d",&b[i][j]);
d = det(a);
 if (d==0.0)
printf("solution does not exists");
else
{
for (i=0; i<3; i++)
for (j=0; j<1; j++)
        t1[i][j] = b[i][j];
 d1= det (t1);
 for(i=0; i<3; i++)
     for (j=1; j<2; j++)
      t2[i][j] = b[i][j-1];
d2=det (t2);
 for (i=0; i<3; i++)
    for (j=2; j<3; j++)
        t3[i][j] = b[i][j-2];
   d3 = det(t3);
printf("\n solution of given system of linear equations
      is :\n");
 printf("x= %0.2f,\t y=%0.2f, \t z=%0.2f",
       d1/d, d2/d, d3/d);
  }
 getch();
}
```

Output:-

Enter coefficient of variables:
1 2 -1
3 1 1
1 -1 2
Enter constants:
-3
4
6
Solution of the given system of linear equations is:
x = 1.0, y = -1.0, z = 2.0.

7. Rank of a Matrix

AIM:

To find the rank of a given 3X3 matrix by reducing to Echelon form using C.

Algorithm:

Step1: Initialize the rank to the number of columns.

Step2: Iterate through each row:

- If the leading element (diagonal element) is not zero, make all elements below it in the same column zero by subtracting appropriate multiples of the row.
- If the leading element is zero, check if there is a row below with a non-zero element in the same column. If so, swap the rows.
- If all elements in the column are zero, reduce the rank by 1 and move to the next column.
- The rank is the number of non-zero rows left after the above operations.

Program :-

```
#include <stdio.h>
#include <conio.h>
#define R 3
#define C 3
void swap (int mat[R][C], int row1, int row2,
           int col)
{
for (int i=0; i<col; i++)
 {
  int temp = mat [row1] [i];
  mat [row1] [i] = mat [row2] [i];
  mat [row2] [i] = temp;
 }
}
int rankofmatrix (int mat [R] [C])
 {
   int rank = C;
for (int row=0; row<rank; row++)
 {
  if ( mat [row] [row])
   {
     for (int col=0; col < R; col++)
       {
      if (col != row)
     {
     double mult = (double) mat [col] [row] / mat [row]
```

```
[row];
for (int i=0; i<rank; i++)
mat[col][i] -= mult * mat[row][i];
}
}
}
else
{
int reduce=1;
for (int i=row+1; i<R; i++)
{
if (mat[i][row])
{
swap (mat, row, i, rank);
reduce =0;
break;
}
}
if (reduce)
{
rank--;
for (int i=0; i<R; i++)
mat[i][row] = mat[i][rank];
}
row--;
}
}
```

```
return rank;
}
Void main()
{
 int mat[3][3], i, j, rank;
 clrscr();
printf("Enter a 3X3 matrix:\n");
for (i=0; i<3; i++)
    for (j=0; j<3; j++)
        scanf("%d", &mat[i][j]);
rank = rankofmatrix(mat);
printf("\n Rank of the matrix is: %d\n", rank);
getch();
}
```

Output :-

Enter a 3x3 matrix :
-1 0 6
3 4 1
-5 1 3
Rank of the matrix is : 3.

8. Arrangement of two-way frequency distribution table.

Problem:

Prepare a two-way frequency distribution for the following data given by ages in years and blood pressure using class intervals (45, 141); (26, 130); (62, 150); (28, 114); (55, 138); (36, 120); (48, 142); (40, 139); (28, 105); (32, 135); (31, 153); (37, 151); (59, 149); (50, 151); (48, 121); (47, 126); (33, 131); (42, 154); (49, 151); (34, 118).

Procedure :-

Let, x = Age in years, y = Blood pressure.

Bivariate frequency distribution is to be prepared by taking class intervals for x as 25-35, 35-45, 45-55, etc., and for y as 105-120, 120-135, etc., using 'four and cross method'.

Calculation:-

Blood pressure (Y) / Age in years (x)	105 - 120	120 - 135	135 - 150	150 - 165	Total fx
25 - 35	III	II	I	I	7
35 - 45		I	I	II	4
45 - 55		II	II	II	6
55 - 65			II	I	3
Total fx	3	5	6	6	20

9. Problem on mean and median.

AIM:

to find mean and median of a list of n values using C.

Algorithm:

Step 1: Start
step 2: read n
step 3: read n values a[i]
step 4: initialize i=1
step 5: if i<=n
goto step 6
else
goto step 10
step 6: initialize j=i+1
step 7: if j<=n goto step 8
else
~~goto~~ set i=i+1 and goto step 5
step 8: if a[i]>a[j]
t = a[i]
a[i] = a[j]
a[j] = t
step 9: set j=j+1 and goto step 7
step 10: set sum=0
step 11: initialize i=1
step 12: if i<=n goto step 13
else goto step 14
step 13: set sum=sum+a[i] and i=i+1

Print mean = sum/n

step 14: print mean = sum/n

step 15: if (n%2 == 0)

print median = (a[n/2] + a[(n/2)+1])/2

else

printf median = a[(n+1)/2]

step 16: Stop.

Program :-

```
#include <stdio.h>
#include <conio.h>
void main ()
{
int n, i, j ;
float a[50], t, sum=0.0;
clrscr();
printf("How many values:\n");
 scanf ("%d",&n);
printf ("Enter %d values:\n",n);
for (i=1; i<=n; i++)
   scanf("%f",&a[i]);
for (i=1; i<=n; i++)
{
 for (j=i+1; j<=n; j++)
  {
    if (a[i]>a[j])
     {
      t=a[i];
      a[i]=a[j];
      a[j]=t;
     }
  }
}
for (i=1; i<=n; i++)
   sum=sum+a[i];
```

```
printf("\n mean = %0.2f", sum/n);
if (n%2==0)
  printf("\n median = %0.2f", (a[n/2]+a[(n/2)+1])/2);
  else
printf("\n median= %0.2f", a[(n+1)/2]);
  getch();
}
```

Output :-

How many values :
6
Enter 6 values :
26
8
12
15
32
6
mean = 16.50
median = 13.50.

10. Empirical relationship between mean, median and mode.

AIM:-

To Establish the empirical relationship among mean, median and mode by using c.

Algorithm:-

Step 1: start
Step 2: print "1. mean 2. median 3. mode".
Step 3: read. choice as d
Step 4: switch (d)
case 1: read median and mode
print mean = (3* median - mode.) / 2
goto step 5
case 2: read mean and mode
print median = (mode + 2* mean) / 3
goto step 5
Case 3: read median and mode
print mode. = 3* median - 2* mean
goto step 5
case default: print "wrong choice" and goto step 2.

Step 5: stop

Program :-

```
# include <stdio.h>
# include <conio.h>
void main()
{
float mean, median, mode;
int d;
clrscr();
nxt: printf("Enter your choice
    \n1. mean \n2. median \n3. mode. \n");
scanf("%d", &d);
switch (d)
  {
  case 1 : printf("Enter median and mode.\n");
           scanf("%f %f", &median, &mode);
           printf("mean = %f", (3*median - mode)/2);
           break;
  case 2 : printf("Enter mean and mode.\n");
           scanf("%f %f", &mean, &mode);
           printf("median = %f", (mode + 2*mean)/3);
           break;
  case 3 : printf("Enter mean and median\n");
           scanf("%f %f", &mean, &median);
           printf("mode = %f", 3*median - 2*mean);
           break;
  default : printf("wrong choice\n");
            goto nxt;
  }
}
```

Output :-

Enter your choice
1. mean
2. median
3. mode
3
Enter mean and median :
22.5
20
mode = 15.00