M.Sc. MATHEMATICS LAB MANUAL

2nd Semester

Prepared By

Pure & Applied Science Dept.

Mathematics

MIDNAPORE CITY COLLEGE

INSTRUCTIONS TO STUDENTS

• Before entering the lab, the student should carry the following things (MANDATORY)

1. Identity card issued by the college.
2. Class notes
3. Lab observation book
4. Lab Manual
5. Lab Record

• Student must sign in and sign out in the register provided when attending the lab session without fail.

• Come to the laboratory in time. Students, who are late more than 10 min., will not be allowed to attend the lab.

• Students need to maintain 80% attendance in lab if not a strict action will be taken.

• All students must follow a Dress Code while in the laboratory.

• Foods, drinks are NOT allowed.

• All bags must be left at the indicated place.

• Refer to the lab staff if you need any help in using the lab.

• Respect the laboratory and its other users.

• Workspace must be kept clean and tidy after experiment is completed.

• Read the Manual carefully before coming to the laboratory and be sure about what you are supposed to do.

• Do the experiments as per the instructions given in the manual.

• Copy all the programs to observation which are taught in class before attending the lab session.

• Students are not supposed to use floppy disks, pen drives without permission of lab- in charge.

• Lab records need to be submitted on or before the date of submission.

C-PROGRAMMING WITH NUMERICAL METHODS LABORATORY MANUAL

(Course Number: MTM 297)

SL.NO	COURSE CONTENT
1	Linear and Binary Search Bubble, Insertion, Selection sort techniques, String and File, Matrices
2	Roots of equation
3	Gauss elimination, Gauss Seidal, Matrix inversion, LU decomposition methods, Solution of Tri-diagonal equations.
4	Lagrange, Newton forward and Backward interpolation, Cubie spline interpolation
5	Gauss quadrature rule, Integration by Monte Carlo method, Double integration.
6	Euler and Modified Euler, Runge-Kutta, Milne-Simpson method
7	Power method and Jacobi method for eigenvalues.

Searing Technique:

1.1 Write a program in C to search a number from a dynamic array of numbers by Linear Search technique.

Program:

```
//Linear Search technique
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
int LinearSearch(int p[],int ele, int no){
int i;
   for(i=0;i<no;i++){
     if(p[i]==ele){
                return i;
               }
     }
     return -1;
}
int main(){
int i,no,ele,result;
int *p;
clrscr();
printf("Enter the number of elements: \n");
scanf("%d",&no);
p=(int*)malloc(no*sizeof(int));
for(i=0;i<no;i++){
scanf("%d",&p[i]);
}
printf("Elements of integer array are: \n");
for(i=0;i<no;i++){
printf(" %d ",p[i]);
}
printf("\n Enter an item to be search using Linear_Search Method: \n");
scanf("%d",&ele);
result=LinearSearch(p,ele,no);
if(result==-1){
     printf("Element is not found: ");
}
else{
     printf("Element is found at position: %d",result);
}
free(p);
getch();
return 0;
}
```

Input and Output Section:

```
Enter the number of elements:
10
10 20 34 56 78 90 12 15 16 17
Elements of integer array are:
 10 20 34 56 78 90 12 15 16 17
 Enter an item to be search using Linear_Search Method:
17
Element is found at position: 9
```

1.2 Write a program in C to search a number from a dynamic sorted array of numbers by Binary Search technique.

Program:

```
//Binary Search technique
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
int BinarySearch(int a[],int ele, int no){
int i,low,high,mid;
low=0;
high=no-1;
while(low<=high){
            mid=(low+high)/2 ;
            if(ele==a[mid]){
                  return mid;
            }
            else if(ele<a[mid]){
                  high=mid-1;
            }
            else
                  low=mid+1;
    }
    return -1;
}
int main(){
int i,no,ele,result;
int *p;
clrscr();
printf("Enter the number of elements: \n");
scanf("%d",&no);
p=(int *)malloc(no*sizeof(int));
for(i=0;i<no;i++){
scanf("%d",&p[i]);
}
printf("Elements of integer array are: \n");
for(i=0;i<no;i++){
printf(" %d ",p[i]);
}
printf("\n Enter an item to be search using Binary_Search Method: \n");
scanf("%d",&ele);
result=BinarySearch(p,ele,no);
if(result==-1){
    printf("Element is not found: ");
}
else{
    printf("Element is found at position: %d",result);
}
free(p);
getch();
return 0;
}
```

Input and Output Section:

```
Enter the number of elements:
6
10 15 19 17 18 20
Elements of integer array are:
 10  15  19  17  18  20
 Enter an item to be search using Binary_Search Method:
19
Element is found at position: 2
```

Sorting Technique:

1.3 Write a program in C to sort a dynamic array of numbers by Selection sort technique.

Program:

```
//Selection Sort
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
void Selection(int a[],int n){
int i,k,j,temp;
for(i=0;i<n-1;i++){
    for(j=k=i;j<n;j++){
                    if(a[j]<a[k]){
                            k=j;
                            }
                    }
                temp=a[i];
                a[i]=a[k];
                a[k]=temp;

    }
}
int main(){
     int i,no;
     int *p;
     clrscr();
    printf("Enter the how many elements to be sort using Selection soring: \n");
     scanf("%d",&no);
     p=(int *)malloc(no*sizeof(int));
     printf("Elements are: \n");
     for(i=0;i<no;i++){
     scanf("%d",&p[i]);
     }
     printf("Before sotring the elements are: \n");
     for(i=0;i<no;i++){
     printf(" %d ",p[i]);
     }
     Selection(p,no);
     printf("\n After Selection sorting the elements are: \n");
```

```
        for(i=0;i<no;i++){
        printf(" %d ",p[i]);
        }
        free(p);
    getch();
    return 0;
    }
```

Input and Output Section:

```
Enter the how many elements to be sort using Selection soring:
8
Elements are:
10 15 14 17 99 78 77 89
Before sotring the elements are:
 10  15  14  17  99  78  77  89
 After Selection sorting the elements are:
 10  14  15  17  77  78  89  99
```

1.4 Write a program in C to sort a dynamic array of numbers by insertion sort technique.

Program:

```
//Insertion Sort
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
void Insertion(int p[],int n){
int i,j,x;
for(i=1;i<n;i++){
    j=i-1;
    x=p[i];
    while(j>-1 && p[j]>x){
            p[j+1]=p[j];
            j--;
            }
     p[j+1]=x;
    }
}
int main(){
     int *p,i,no;
     clrscr();
    printf("Enter the how many elements to be sort using Insertion soring: \n");
     scanf("%d",&no);
     p=(int *)malloc(no*sizeof(int));
     printf("Elements are: \n");
     for(i=0;i<no;i++){
     scanf("%d",&p[i]);
     }
     printf("Before sotring the elements are: \n");
     for(i=0;i<no;i++){
     printf(" %d ",p[i]);
     }
     Insertion(p,no);
     printf("\n After Insertion sorting the elements are: \n");
     for(i=0;i<no;i++){
     printf(" %d ",p[i]);
     }
getch();
free(p);
return 0;
}
```

Input and Output Section:

```
Enter the how many elements to be sort using Insertion soring:
5
Elements are:
89 67 545 63 11
Before sotring the elements are:
 89  67  545  63  11
 After Insertion sorting the elements are:
 11  63  67  89  545
```

1.5 Write a program in C to sort a dynamic array of numbers by bubble sort technique.

Program:

```
//Bubble Sort
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
void Bubble(int a[],int n){
int i,j,temp;
for(i=0;i<n-1;i++){
    for(j=0;j<n-i-1;j++){
            if(a[j]>a[j+1]){
                    temp=a[j];
                    a[j]=a[j+1];
                    a[j+1]=temp;
                   }
            }
    }
}
int main(){
     int *p,i,no;
     clrscr();
     printf("Enter the how many elements to be sort using Bubble soring: \n");
     scanf("%d",&no);
     p=(int *)malloc(no*sizeof(int));
     printf("Elements are: \n");
     for(i=0;i<no;i++){
     scanf("%d",&p[i]);
     }
     printf("Before sorting the elements are: \n");
     for(i=0;i<no;i++){
     printf(" %d ",p[i]);
     }
     Bubble(p,no);
     printf("\n After Bubble sorting the elements are: \n");
     for(i=0;i<no;i++){
     printf(" %d ",p[i]);
     }
     free(p);
getch();
return 0;
}
```

Input and Output Section:

```
Enter the how many elements to be sort using Bubble soring:
5
Elements are:
10 38 56 8 56
Before sorting the elements are:
 10  38  56  8  56
 After Bubble sorting the elements are:
 8  10  38  56  56
```

String and File

1.6 Write a program in C to check a string is a palindrome or not using user defined function.

Program:

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
int CheckPalindrom(char str[]){
int length,i,j,flag=0;
length=strlen(str);
for(i=0,j=length-1;i<length;i++,j--){
        if(str[i]!=str[j]){
                        flag++;
                }
        }
        return flag;

}
int main()
{
char str[10];
int flag;
//clrscr();
printf("Enter the string(word): \n");
//scanf("%s",str);
gets(str);
flag=CheckPalindrom(str);
if(flag==0)
printf("%s is palindrome ",str);
else
printf("%s is not palindrome ",str);
getch();
return 0;
}
```

Input and Output Section:

```
Enter the string(word):
madam
madam is palindrome

Enter the string(word):
madaM
madaM is not palindrome
```

1.7 Write a program in C to convert the letter contains in given text file as lowercase to uppercase.

Input Section:

Input.txt file:

wellcome to midnapore city college.

Program:

```
#include<stdio.h>
#include<stdlib.h>
int main(){
    FILE *fp1,*fp2;
    char ch;
    fp1=fopen("Input.txt","r");
    fp2=fopen("Duplicate.txt","w");
    if((fp1==NULL) || (fp2==NULL)){
            printf("File does not exit ");
    }
    ch=fgetc(fp1);
    while(ch!=EOF){
            if(ch>=97 && ch<=122){
                    ch=ch-32;
                    fputc(ch,fp2);
            }
            else{
                    fputc(ch,fp2);
            }
            ch=fgetc(fp1);
    }
    fclose(fp1);
    fclose(fp2);
    remove("Input.txt");
    rename("Duplicate.txt","Input.txt");
    return 0;
}
```

Output Section:

```
Input.txt file:
WELLCOME TO MIDNAPORE CITY COLLEGE.
```

1.8 Write a program in C to store the records of the students in a file.

Program:

```
#include<stdio.h>
#include<stdlib.h>
#include<errno.h>
int main(){
int n,i;
char name[50];
int marks;
FILE *fp=fopen("Student.txt","w");
if(fp==NULL){
    printf("Error value is %d \n",errno);
    printf("The error message is %s \n",strerror(errno));
    exit(EXIT_FAILURE);
}
    printf("Enter the number of student: \n");
    scanf("%d",&n);
    for(i=0;i<n;i++){
            printf("Enter the name of student: ");
            scanf("%s",name);
            printf("Enter the marks: ");
            scanf("%d",&marks);
            fprintf(fp, "Name: %s | Marks: %d \n",name,marks);
                    }
    fclose(fp);
    return 0;
}
```

Input and Output Section:

```
Enter the number of student:
5
Enter the name of student: Soumen
Enter the marks: 65
Enter the name of student: Radha
Enter the marks: 90
Enter the name of student: Atanu
Enter the marks: 78
Enter the name of student: Raghu
Enter the marks: 15
Enter the name of student: Sanjoy
Enter the marks: 88
```

Show Student.txt file

```
Name: Soumen | Marks: 65
Name: Radha | Marks: 90
Name: Atanu | Marks: 78
Name: Raghu | Marks: 15
Name: Sanjoy | Marks: 88
```

Matrices

1.9 Write a program in C to create two matrices using dynamic memory allocation and perform addition and subtraction operation among them.

Program:

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
void MatrixAddition(int **a, int r1, int c1, int **b,int r2,int c2){
    int **add,i,j;
    //Result matrix add dynamic memory allocation
    add=(int **)malloc(r1*sizeof(int *));
    for(i=0;i<r1;i++){
            add[i]=(int *)malloc(c2*sizeof(int));
    }
    for(i=0;i<r1;i++){
            for(j=0;j<c2;j++){
                    add[i][j]=a[i][j]+b[i][j];
            }
    }
    //Addition Matrix show
    printf("Addition of two matrix is: \n");
    for(i=0;i<r1;i++){
            for(j=0;j<c1;j++){
                    printf("%d ",add[i][j]);
            }
            printf("\n");
    }
    free(add);
}
void MatrixSubtraction(int **a, int r1, int c1, int **b,int r2,int c2){
    int **sub,i,j;
    //Result matrix sub dynamic memory allocation
    sub=(int **)malloc(r1*sizeof(int *));
    for(i=0;i<r1;i++){
            sub[i]=(int *)malloc(c2*sizeof(int));
```

```
     }
     for(i=0;i<r1;i++){
               for(j=0;j<c2;j++){
                         sub[i][j]=a[i][j]-b[i][j];
               }
     }
     //Subtraction Matrix show
     printf("Subtraction of two matrix is: \n");
     for(i=0;i<r1;i++){
               for(j=0;j<c1;j++){
                         printf("%d ",sub[i][j]);
               }
               printf("\n");
     }
     free(sub);
}
int main(){
     int **a,r1,c1,**b,r2,c2,i,j;
     printf("Enter rows and columns of first matrix: \n");
    scanf("%d%d",&r1,&c1);

    // A Matrix dynamic memory allocation
    a=(int**)malloc(r1*sizeof(int*));
    for(i=0;i<r1;i++){
     a[i]=(int *)malloc(c1*sizeof(int));
     }
    printf("Enter elements of first matrix: \n");
    for(i=0;i<r1;i++){
     for(j=0;j<c1;j++){
               scanf("%d",&a[i][j]);
               }
     }
    printf("\nThe first matrix is: \n");
     for(i=0;i<r1;i++){
     for(j=0;j<c1;j++){
               printf("%d ",a[i][j]);
               }
```

```
            printf("\n");
     }
     printf("Enter rows and columns of second matrix: \n");
    scanf("%d%d",&r2,&c2);

    // B Matrix dynamic memory allocation
    b=(int**)malloc(r2*sizeof(int*));
    for(i=0;i<r2;i++){
     b[i]=(int *)malloc(c2*sizeof(int));
     }
    printf("Enter elements of second matrix: \n");
    for(i=0;i<r2;i++){
     for(j=0;j<c2;j++){
            scanf("%d",&b[i][j]);
            }
     }
    printf("\nThe second matrix is: \n");
     for(i=0;i<r2;i++){
     for(j=0;j<c2;j++){
            printf("%d ",b[i][j]);
            }
            printf("\n");
     }
     MatrixAddition(a,r1,c1,b,r2,c2);
     MatrixSubtraction(a,r1,c1,b,r2,c2);
     free(a);
     free(b);
    getch();
     return 0;
}
```

Input and Output Section:

```
Enter rows and columns of first matrix:
3 3
Enter elements of first matrix:
3 4 5
6 7 8
```

```
-10 -11 -12

The first matrix is:
3 4 5
6 7 8
-10 -11 -12
Enter rows and columns of second matrix:
3 3
Enter elements of second matrix:
-1 -2 -3
4 12 1
19 9 3

The second matrix is:
-1 -2 -3
4 12 1
19 9 3
Addition of two matrix is:
2 2 2
10 19 9
9 -2 -9
Subtraction of two matrix is:
4 6 8
2 -5 7
-29 -20 -15
```

1.10 Write a program in C to create two matrices using dynamic memory allocation and perform multiplication and transpose operations among them.

Program:

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
void MatrixMultiplication(int **a,int **b,int **c,int r1,int c1,int r2,int c2)
{
   int i,j,k;
   for(i=0;i<r1;i++)
   {
      for(j=0;j<c2;j++)
      {
      c[i][j]=0;
         for(k=0;k<c1;k++){
               c[i][j]=c[i][j]+a[i][k] * b[k][j];
                       }

               }
   }
   //Multiplication Matrix show
   printf("\nThe product is: \n");
   for(i=0;i<r1;i++){
        for(j=0;j<c2;j++){
               printf("%d ",c[i][j]);
               }
               printf("\n");
        }
}
void TransposeMatrix(int **a,int r1,int c1){
        int i,j;
        printf("\nThe Transpose of Matrix A is: \n");
        for(i=0;i<c1;i++){
               for(j=0;j<r1;j++){
                       printf("%4d",a[j][i]);
               }
               printf("\n");
        }
}
int main()
{
```

```
int **a,r1,c1,**b,r2,c2,**c,i,j;
printf("Enter rows and columns of first matrix: \n");
scanf("%d%d",&r1,&c1);
// A Matrix dynamic memory allocation
a=(int**)malloc(r1*sizeof(int*));
for(i=0;i<r1;i++){
      a[i]=(int *)malloc(c1*sizeof(int));
      }

printf("Enter elements of first matrix: \n");
for(i=0;i<r1;i++){
      for(j=0;j<c1;j++){
               scanf("%d",&a[i][j]);
               }
      }
printf("\nThe first matrix is: \n");
      for(i=0;i<r1;i++){
      for(j=0;j<c1;j++){
               printf("%d ",a[i][j]);
               }
               printf("\n");
      }
printf("Enter rows and columns of second matrix: \n");
scanf("%d%d",&r2,&c2);
      // B Matrix dynamic memory allocation
b=(int**)malloc(r2*sizeof(int*));
for(i=0;i<r2;i++){
      b[i]=(int *)malloc(c2*sizeof(int));
      }
printf("Enter elements of second matrix: \n");
      for(i=0;i<r2;i++){
      for(j=0;j<c2;j++){
               scanf("%d",&b[i][j]);
               }
      }
printf("\nThe second matrix is: \n");
      for(i=0;i<r2;i++){
      for(j=0;j<c2;j++){
               printf("%d ",b[i][j]);
               }
               printf("\n");
      }
if(c1!=r2)
```

```
   {
      printf("\nInvalid dimensions.");
      //exit(0);
   }
   else{
        // Result C Matrix dynamic memory allocation
        c=(int**)malloc(sizeof(int*)*r1);
        for(i=0;i<r1;i++){
                        c[i]=(int *)malloc(c2*sizeof(int));
                }
   //Matrix Function Call
   MatrixMultiplication(a,b,c,r1,c1,r2,c2);
        }
   //Transpose function call
        TransposeMatrix(a,r1,c1);
   free(a);
   free(b);
   free(c);
   getch();
   return 0;
}
```

Input and Output Section:

```
Enter rows and columns of first matrix:
2 3
Enter elements of first matrix:
2 3 4
5 6 7

The first matrix is:
2 3 4
5 6 7
Enter rows and columns of second matrix:
3 2
Enter elements of second matrix:
2 2
2 5
6 7
```

```
The second matrix is:
2 2
2 5
6 7
The product is:
34 47
64 89

The Transpose of Matrix A is:
   2   5
   3   6
   4   7
```

Roots of Equation:

2.1 Write a program in C to find a real root of an equation x^3-8x-4=0 by Regula-Falsi method.

Solution:

Let f(x)= x^3-8x-4

f(0)=-4	i.e. (-ve)
f(1)=-11	i.e (-ve)
f(2)=-12	i.e. (-ve)
f (3) =-1	i.e. (-ve)
f (4) =28	i.e. (+ve)

Hence the root lies between 3 and 4.

Program:

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
float Regula_falsi(float x){
return x*x*x-8*x-4;
}
int main(){
float a,b,x,eps;
clrscr();
printf("Enter the value of a,b,eps \n");
scanf("%f%f%f",&a,&b,&eps);
while(fabs (Regula_falsi(b) )> eps)
{
x=(a*Regula_falsi(b)-b*Regula_falsi(a))/(Regula_falsi(b)-Regula_falsi(a));
    if(Regula_falsi(a)*Regula_falsi(x)>0)
            b=x;
    else
            a=x;
}
    printf("The real root is =%f",x);
    getch();
return 0;
}
```

Input and Output Section:

```
Enter the value of a,b,eps
3 4 0.0001
The real root is =3.051374
```

2.2 Write a program in C to find a real root of an equation x^3-8x-4=0 which lies between 3 and 4, by Newton-Raphson Method.

Solution:

Let f(x)= x^3-8x-4

f '(x)=$3x^2-8$

f (3) =-1 i.e. (-ve)

f (4) =28 i.e. (+ve)

Hence the root lies between 3 and 4.

Let the initial guess value x0=3.

Program:

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
float f(float x){
    return x*x*x-8*x-4.0;
}
float df(float x){
return 3.0*x*x-8.0;
}
int main(){
        float x0,x1,q,eps;
        int k=1;
        printf("Enter the initial guess value and eps \n");
        scanf("%f%f",&x0,&eps);
        do{
        q=x0;
        x1=x0-(f(x0)/df(x0));
        x0=x1;
        k=k+1;
        }while(fabs(x1-q)>eps);
        printf("Real root is=%f \n",x1);
        printf("Number of steps=%d",k);
getch();
return 0;
}
```

Input and Output Section:

```
Enter the initial guess value and eps
3 0.00001
Real root is=3.051374
Number of steps=4
```

2.3 Compute the root of the equation x^3-4x-9=0 correct to 3 decimal places, using Bisection method.

Solution:

Let f(x)= x^3-4x-9

f (1) =1-4-9=-12 i.e. (-ve)

f (2) =8-8-9=-9 i.e. (-ve)

f (3) =27-12-9=6 i.e. (+ve)

Hence the root lies between 2 and 3.

Program:

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
float Bisection(float x){
return (pow(x,3)-4*x-9);
}
int main(){
        float a,b,x,eps;
        clrscr();
        printf("Enter the value of a,b and eps \n");
        scanf("%f%f%f",&a,&b,&eps);
        x=(a+b)/2;
        while(fabs(x-b)>eps){
        if(Bisection(a)*Bisection(x)>0)
                a=x;
         else
                b=x;
        // printf("The middle point = %f",x);
         x=(a+b)/2;
```

```
 }
          printf("The approximate root is =%.3f",x);
          getch();
return 0;
}
```

Input and Output Section:

```
Enter the value of a, b and eps
2  3   0.00001
The approximate root is =2.707
```

***2.4 Find the real root of the equation x^3-x-1=0 using iteration method*.**

Program:

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
float f(float x){
return pow(((1+x)/x),.5);
}
int main(){
float x1,x2,r;
clrscr();
printf("Enter the initial guess value \n");
scanf("%f",&x1);
x2=f(x1);
do{
x1=x2;
x2=f(x1);
r=fabs((x2-x1)/x2);
}while(r>0.001);
printf("The real root=%f",x2);
getch();
return 0;
}
```

Input and Output Section:

```
Enter the initial guess value
1
The real root=1.324901
```

System of Linear Equation

3.1 Write a program to solve following system of equation by Gauss-elimination method:

x+2y+3z=7

2x+7y+15z=26

3x+15y+41z=26

Solution:

Augmented Matrix:

$$\begin{bmatrix} 1 & 2 & 3 \ldots\ldots .7 \\ 2 & 7 & 15 \ldots . .26 \\ 3 & 15 & 41 \ldots . .26 \end{bmatrix}$$

Program:

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
int main()
{
   int i,j,k,n;
   float A[20][20],c,x[10],sum=0.0;
   clrscr();
   printf("\nEnter the order of matrix: ");
   scanf("%d",&n);
   printf("\nEnter the elements of augmented matrix row-wise:\n\n");
   for(i=1; i<=n; i++)
   {
        for(j=1; j<=(n+1); j++)
        {
           printf("A[%d][%d] : ", i,j);
           scanf("%f",&A[i][j]);
        }
   }
   /* loop for the generation of upper triangular matrix*/
   for(j=1; j<=n; j++)
   {
        for(i=1; i<=n; i++)
        {
           if(i>j)
           {
                c=A[i][j]/A[j][j];
                for(k=1; k<=n+1; k++)
                {
                   A[i][k]=A[i][k]-c*A[j][k];
                }
           }
        }
    }
    x[n]=A[n][n+1]/A[n][n];
    /* this loop is for backward substitution*/
    for(i=n-1; i>=1; i--)
    {
        sum=0;
        for(j=i+1; j<=n; j++)
        {
           sum=sum+A[i][j]*x[j];
        }
        x[i]=(A[i][n+1]-sum)/A[i][i];
    }
    printf("\nThe solution is: \n");
    for(i=1; i<=n; i++)
    {
        printf("\nx%d=%.2f\t",i,x[i]);
    }
    getch();
    return(0);
}
```

Input and Output Section:

```
Enter the order of matrix: 3
Enter the elements of augmented matrix row-wise:
A[1][1] : 1
A[1][2] : 2
A[1][3] : 3
A[1][4] : 7
A[2][1] : 2
A[2][2] : 7
A[2][3] : 15
A[2][4] : 26
A[3][1] : 3
A[3][2] : 15
A[3][3] : 41
A[3][4] : 26
The solution is:
x1=-19.60
x2=22.60
x3=-6.20
```

3.2 Write a program in C to find the solutions of a system of linear equations

$$-3x_1 + x_2 - 5x_3 = -12$$

$$x_1 + 2x_2 + 4x_3 = 11$$

$$x_2 + 2x_3 = 5$$

by Gauss-Seidal method.

Program:

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#define eps 0.0001
#define x1(x2,x3) ((12+x2-5*x3)/3.0)
#define x2(x1,x3) ((11-x1-4*x3)/2.0)
#define x3(x1,x2)  ((5-x2)/2.0)
int main(){
float x1=0,x2=0,x3=0,y1,y2,y3;
int flag=0;
clrscr();
printf("\n \t x1 \t\t x2 \t\t x3");
printf("\n \t %f \t %f \t %f",x1,x2,x3);
do{
          y1=x1(x2,x3);
          y2=x2(y1,x3);
          y3=x3(y1,y2);
      if(fabs(x1-y1)<eps && fabs(x2-y2)<eps && fabs(x3-y3)<eps){
          printf("\n x1=%.3f",y1);
          printf("\n x2=%.3f",y2);
          printf("\n x3=%.3f",y3);
            flag=1;
      }
     else{
       x1=y1;
       x2=y2;
       x3=y3;
     printf("\n \t %f \t %f \t %f",x1,x2,x3);
     }
} while(flag!=1);
getch();
return 0;
}
```

Input and Output Section:

```
         x1              x2              x3
        0.000000        0.000000        0.000000
        4.000000        3.500000        0.750000
        3.916667        2.041667        1.479167
        2.215278        1.434028        1.782986
        1.506366        1.180845        1.909578
        1.210986        1.075352        1.962324
        1.087911        1.031397        1.984302
        1.036629        1.013082        1.993459
        1.015262        1.005451        1.997275
        1.006359        1.002271        1.998864
        1.002650        1.000946        1.999527
        1.001104        1.000394        1.999803
        1.000460        1.000164        1.999918
        1.000192        1.000068        1.999966
        1.000080        1.000028        1.999986
```

```
 x1=1.000
 x2=1.000
 x3=2.000
```

3.3 Write a program in C to find the solutions of a system of linear equations

$$-3x_1 + x_2 - 5x_3 = -12$$

$$x_1 + 2x_2 + 4x_3=11$$

$$x_2 + 2x_3=5$$

by LU decomposition method.

Program:

```
#include<stdio.h>
#include<conio.h>
int main(){
	float a[10][10],l[10][10],u[10][10],z[10],x[10],b[10];
	int i,j,k,n;
	printf("Enter the size of the coefficient matrix  \n");
	scanf("%d",&n);
	printf("Enter the elements rowwise \n");
	for(i=1;i<=n;i++){
		for(j=1;j<=n;j++){
			scanf("%f",&a[i][j]);
		}

	}
	printf("Enter the right hand vector \n");
	for(i=1;i<=n;i++){
		scanf("%f",&b[i]);
	}

		//Computations of L and U matrices
	for(i=1;i<=n;i++){
		l[i][1]=a[i][1];
	}
	for(j=2;j<=n;j++){
		u[1][j]=a[1][j]/l[1][1];
	}
	for(i=1;i<=n;i++){
        u[i][i]=1;
}
for(i=2;i<=n;i++){
        for(j=2;j<=n;j++){
                if(i>=j){
                        l[i][j]=a[i][j];
                        for(k=1;k<=j-1;k++){
                                l[i][j]=l[i][j]-l[i][k]*u[k][j];
                        }
                }
                else{
                        u[i][j]=a[i][j];
                        for(k=1;k<=i-1;k++){
                                u[i][j]=u[i][j]-l[i][k]*u[k][j];
                                u[i][j]=u[i][j]/l[i][i];
                        }
                }
        }
}
printf("The lower triangular matrix L \n");
for(i=1;i<=n;i++){
        for(j=1;j<=i;j++){
                printf("%f ",l[i][j]);
                        }
                printf("\n");
}
printf("The Upper triangular matrix U \n");
for(i=1;i<=n;i++){
        for(j=1;j<=i;j++){
                printf(" ");
                for(j=i;j<=n;j++){
                        printf("%f ",u[i][j]);
                }
                printf("\n");
        }
}

/*solve Lz=b by forward substituion */
z[1]=b[1]/l[1][1];
for(i=2;i<=n;i++){
```

```
                z[i]=b[i];
                for(j=1;j<=i-1;j++){
                        z[i]=z[i]-l[i][j]*z[j];
                }
                        z[i]=z[i]/l[i][i];
        }

        /*solve Ux=z by backward substitution*/
        x[n]=z[n];
        for(i=n-1;i>=1;i--){
                x[i]=z[i];
                for(j=i+1;j<=n;j++){
                        x[i]=x[i]-u[i][j]*x[j];
                }
        }

        printf("The solution is: \n");
        for(i=1;i<=n;i++){
                printf("%f ",x[i]);
        }
        getch();
        return 0;
}
```

Input and Output Section:

```
Enter the size of the coefficient matrix
3
Enter the elements rowwise
-3 1 -5
1 2 4
0 1 2
Enter the right hand vector
-12 11 5
The lower triangular matrix L
-3.000000
1.000000 2.333333
0.000000 1.000000 1.000000
The Upper triangular matrix U
 1.000000 -0.333333 1.666667
 1.000000 1.000000
          1.000000
The solution is:
0.999999 1.000000 2.000000
```

3.4 Write a program in C to find the solution of a Tridiagonal system of equations

$$\begin{aligned} x_1 + x_2 \quad &= 3 \\ x_1 + x_2 - 3x_3 &= -3 \\ -x_2 + 3x_3 &= 4 \end{aligned}$$

Program:

```
#include<stdio.h>
#include<stdlib.h>
float x[10]; //x[i] is the solution of the tridiagonal system of equation
float TriDiag(float a[10],float b[10],float c[10],float d[10],int n){
	int i;
	float gamma[10],z[10];
	gamma[1]=b[1];
	for(i=2;i<=n;i++){
		if(gamma[i-1]==0.0){
			printf("A minor is zero: Method falis \n");
			exit(0);
		}
		gamma[i]=b[i]-a[i]*c[i-1]/gamma[i-1];
	}
	z[1]=d[1]/b[1];
	for(i=2;i<=n;i++){
		z[i]=(d[i]-a[i]*z[i-1])/gamma[i];
}
```

```
                x[n]=z[n];
                for(i=n-1;i>=1;i--){
                        x[i]=z[i]-c[i]*x[i+1]/gamma[i];
                }

        return x[0];
}
int main(){
        float a[10],b[10],c[10],d[10];
        int i,n;
        float y;
        printf("Enter the size of the coefficient matrix : \n");
        scanf("%d",&n);
        printf("Enter the first row (only non zero elements) \n");
        scanf("%f%f",&b[1],&c[1]);
        printf("Enter the row 2 to n-1 :\n");
        for(i=2;i<=n-1;i++){
                scanf("%f%f%f",&a[i],&b[i],&c[i]);
        }
        printf("Enter the last row : \n");
        scanf("%f%f",&a[n],&b[n]);
        printf("Enter the right hand vector : \n");
        for(i=1;i<=n;i++){
                scanf("%f",&d[i]);
        }
        y=TriDiag(a,b,c,d,n);
        printf("The solution is: \n");
        for(i=1;i<=n;i++){
                printf("%f ",x[i]);
        }
        return 0;
}
```

Input and Output Section:

```
Enter the size of the coefficient matrix :
3
Enter the first row (only non zero elements)
1 1
Enter the row 2 to n-1 :
1 1 -3
Enter the last row :
-2 3
Enter the right hand vector :
3 -3 4
A minor is zero: Method falis
```

Solve the following tri-diagonal system of equation.

x1+x2=3

-x1+2x2+x3=6

3x2+2x3=12

```
Enter the size of the coefficient matrix :
3
Enter the first row (only non zero elements)
1 1
Enter the row 2 to n-1 :
-1 2 1
Enter the last row :
3 2
Enter the right hand vector :
3 6 12
The solution is:
1.000000 2.000000 3.000000
```

Interpolation

4.1 Write a program in C to find the value of a function f(x) using given tabular values by Lagrange Interpolation method. Test the program to find f(2.18) using the following:

x	2	2.2	2.4	2.6	2.8	3
f(x)	0.30103	0.34242	0.38041	0.41497	0.44716	0.47721

Program:

```
#include<stdio.h>
#include<conio.h>
int main(){
float x[10],y[10],u,p,sum=0;
int i,j,n;
clrscr();
printf("Enter the how many points \n");
scanf("%d",&n);
printf("Enter the x and y values \n");
    for(i=0;i<n;i++){
         scanf("%f%f",&x[i],&y[i]);
                }
printf("Which value is to be computed \n");
scanf("%f",&u);
for(i=0;i<n;i++){
     p=y[i];
     for(j=0;j<n;j++){
         if(i!=j){
             p=p*(u-x[j])/(x[i]-x[j]);
                }
             }
         sum=sum+p;
        }
printf("The result y(%.2f)= %f",u,sum);
getch();
return 0;
}
```

Input and Output Section:

```
Enter the how many points
6
Enter the x and y values
2       0.30103
2.2     0.34242
2.4     0.38041
2.6     0.41497
2.8     0.44716
3       0.47721
Which value is to be computed
2.18
The result y(2.18)=0.338414
```

4.2 Evaluate f (6) from the table using Newton backword formula.

x	0	1	2	3	4	5
f(x)	1	5	31	121	341	781

Program:

```
#include<stdio.h>
#include<conio.h>
int main(){
        float x[10],y[20],s,u,sum,p=1,h;
        int i,j,n;
        clrscr();
        printf("Enter the how many point \n");
        scanf("%d",&n);
        printf("Enter the value of x and y \n");
        for(i=1;i<=n;i++)
        scanf("%f%f",&x[i],&y[i]);
        h=x[2]-x[1];
        printf("Enter which value is to be computed \n");
        scanf("%f",&u);
        s=(u-x[n])/h;
        sum=y[n];
        for(i=1;i<=n-1;i++){
                for(j=1;j<=n-i;j++)
                {
                        y[j]=y[j+1]-y[j];
                }
                p=p*(s+i-1)/i;
                sum=sum+p*y[j-1];
        }
        printf("The result=%f",sum);
        getch();
return 0;
}
```

Input and Output Section:

```
Enter the how many point
6
Enter the value of x and y
0 1
1 5
2 31
3 121
4 341
5 781
Enter which value is to be computed
6
The result=1555.000000
```

4.3 Evaluate f (2.5) from the table using Newton forward formula.

x	0	1	2	3	4	5
f(x)	41	43	47	53	61	71

Program:

```
#include<stdio.h>
#include<conio.h>
int main(){
float x[10],y[20],s,u,sum,p=1,h;
int i,j,n;
clrscr();
printf("Enter the how many point \n");
scanf("%d",&n);
printf("Enter the value of x and y \n");
for(i=1;i<=n;i++)
        scanf("%f%f",&x[i],&y[i]);
h=x[2]-x[1];
printf("Enter which value is to be computed \n");
scanf("%f",&u);
s=(u-x[1])/h;
sum=y[1];
for(i=1;i<=n-1;i++){
        for(j=1;j<=n-i;j++)
                 {
                         y[j]=y[j+1]-y[j];
                 }
                p=p*(s-i+1)/i;
                sum=sum+p*y[1];
         }
printf("The result=%f",sum);
getch();
return 0;
}
```

Input and Output Section:

```
Enter the how many point
6
Enter the value of x and y
0 41
1 43
2 47
3 53
4 61
5 71
Enter which value is to be computed
2.5
The result=49.750000
```

4.4 Write a program in C to find the natural Cubic spline interpolation for the following information:

x	0.15	0.17	0.18	0.21	0.23
y	0.14944	0.16918	0.18886	0.20846	0.22798

Program:

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#include<stdlib.h>
float M[20];
float TriDiag(float a[20],float b[20],float c[20],float d[20],int n){
        int i;
        float gamma[10],z[10];
        gamma[1]=b[1];
        for(i=2;i<=n;i++){
                if(gamma[i-1]==0.0){
                        printf("A minor is zero: Method fails ");
                        exit(0);
                }
                gamma[i]=b[i]-a[i]*c[i-1]/gamma[i-1];
        }
        z[1]=d[1]/gamma[1];
        for(i=2;i<=n;i++){
                z[i]=(d[i]-a[i]*z[i-1])/gamma[i];
                        }
                        //Computation of M
                        M[n]=z[n];
                        for(i=n-1;i>=1;i--){
                                M[i]=z[i]-c[i]*M[i+1]/gamma[i];
                        }
                        return (M[0]);
}
int main(){
        int i,n;
        char opt,s[10];
        float x[20],y[20],h[20],A[20],B[20],C[20],D[20];
        float a[20],b[20],c[20],d[20],xg,yd0,ydn,temp,yc;
        printf("Enter the subintervals: \n");
```

```
scanf("%d",&n);
printf("Enter x and y values: \n");
for(i=0;i<=n;i++){
        scanf("%f%f",&x[i],&y[i]);
}
printf("Enter interpolation point x: \n");
scanf("%f",&xg);
printf("The given values of x and y are \n x_value  y_value \n");
for(i=0;i<=n;i++){
        printf("%f   %f \n",x[i],y[i]);
}
//computation of h[i]
for(i=0;i<=n;i++){
        h[i]=x[i]-x[i-1];
}
//Computation of A,B,C,D
for(i=1;i<n;i++){
        A[i]=h[i];
        B[i]=2*(h[i]+h[i+1]);
        C[i]=h[i+1];
        D[i]=6*((y[i+1])-y[i])/h[i+1]-(y[i]-y[i-1])/h[i];
}
printf("\n N Natural spline \n");
printf("P Non-Periodic spline \n");
printf("E Extrapolated spline \n");
printf("C End point Curvature adjusted spline \n");
printf("Enter the choice : ");
opt=getche();
switch(toupper(opt)){
        case 'N':
                //Natural spline
                temp=TriDiag(A,B,C,D,n-1);
                M[0]=0;
                M[n]=0;
                break;
        case 'P':
        //Non-periodic spline
        printf("\n Enter the values of y[0] and y[n] ");
        scanf("%f%f",&yd0,&ydn);
        D[0]=6*((y[1]-y[0])/h[1]-yd0)/h[1];
        D[n]=6*(ydn-(y[n]-y[n-1])/h[n])/h[n];
        for(i=n+1;i>=1;i--){
                D[i]=D[i-1];
        }
                A[n+1]=1;
                B[n+1]=2;
                for(i=n;i>=2;i--){
                A[i]=A[i-1];
                B[i]=B[i-1];
                C[i]=C[i-1];
}
B[1]=2;
C[1]=1;
temp=TriDiag(A,B,C,D,n+1);
for(i=0;i<=n;i++){
        M[i]=M[i+1];
        break;
}
case 'E':
        //Extrapolated spline
        B[1]=A[1]+B[1]+A[1]*h[1]/h[2];
        C[1]=C[1]-A[1]*h[1]/h[2];
        A[n-1]=A[n-1]-C[n-1]*h[n]/h[n-1];
        B[n-1]=B[n-1]+C[n-1]+C[n-1]*h[n]/h[n-1];
        temp=TriDiag(A,B,C,D,n-1);
        M[0]=M[1]-h[1]*(M[2]-M[1])/h[2];
        M[n]=M[n-1]+h[n]*(M[n-1]-M[n-2])/h[n-1];
        break;

case 'C':
        //End point Curvature adjusted spline
        printf("\n Enter the values of y[0] and y[n] ");
        scanf("%f%f",&yd0,&ydn);
        D[1]=D[1]-A[1]*yd0;
        D[n-1]=D[n-1]-C[n-1]*ydn;
        temp=TriDiag(A,B,C,D,n-1);
        M[0]=yd0;
        M[n]=ydn;
        break;
default: printf("\n No choice \n");
```

```
        exit(0);
        }
        //Computation of the coefficients of the splines
        for(i=0;i<=n-1;i++){
                a[i]=(M[i+1]-M[i])/(6*h[i+1]);
                b[i]=M[i]/2;
                c[i]=(y[i+1]-y[i])/h[i+1]-(2*h[i+1]*M[i]+h[i+1]*M[i+1])/6;
                d[i]=y[i];
        }
        //printing of splines
        printf("\n The cubic splines are: \n");
        for(i=0;i<n;i++){
                s[1]=(x[i]>0) ? '-':'+';
                s[2]=(b[i]<0) ? '-':'+';
                s[3]=(c[i]<0) ? '-':'+';
                s[4]=(d[i]<0) ? '-':'+';
                temp=fabs(x[i]);
        printf("p%1d(x)=%7.4f(x%c%7.4f)^3%c%7.4f(x%c%7.4f)^2%c%7.4f(x
%c%7.4f)%c%4.7f
\n",i,a[i],s[1],temp,s[2],fabs(b[i]),s[1],temp,s[3],fabs(c[i]),s[1],temp,s[4],fabs(d[
i]));
                printf("      in [%7.4f,%7.4f] \n",x[i],x[i+1]);
        }
        if((xg<x[0]) || (xg>x[n])){
                printf("\n x outside the range ");
                exit(0);
        }
        for(i=0;i<=n-1;i++){
                if(xg<x[i+1]){
                        temp=xg-x[i];
                        yc=a[i]*temp*temp*temp+b[i]*temp*temp+c[i]*temp+d[i];
                        printf("The value of y at x=%f is %f ",xg,yc);
                        exit(0);
                }
        }
        return 0;
}
```

Input and Output Section:

```
0.21 0.20846
0.23 0.22798
Enter interpolation point x:
0.22
The given values of x and y are
 x_value  y_value
0.150000   0.149440
0.170000   0.169180
0.180000   0.188860
0.210000   0.208460
0.230000   0.227980

 N Natural spline
P Non-Periodic spline
E Extrapolated spline
C End point Curvature adjusted spline
Enter the choice : n
 The cubic splines are:
p0(x)=1531.2067(x- 0.1500)^3+ 0.0000(x- 0.1500)^2+ 0.3745(x-
0.1500)+0.1494400
      in [ 0.1500, 0.1700]
p1(x)=-3401.9045(x- 0.1700)^3+91.8724(x- 0.1700)^2+ 1.3895(x-
0.1700)+0.1691800
      in [ 0.1700, 0.1800]
p2(x)=436.1509(x- 0.1800)^3-10.1848(x- 0.1800)^2+ 0.5663(x-
0.1800)+0.1888600
      in [ 0.1800, 0.2100]
p3(x)=-484.4790(x- 0.2100)^3+29.0688(x- 0.2100)^2+ 0.5884(x-
0.2100)+0.2084600
      in [ 0.2100, 0.2300]
The value of y at x=0.220000 is 0.216767
```

Another I/O:

```
Enter the subintervals:
4
Enter x and y values:
0.150000   0.149440
```

```
0.170000   0.169180
0.180000   0.188860
0.210000   0.208460
0.230000   0.227980
Enter interpolation point x:
0.22
The given values of x and y are
 x_value  y_value
0.150000   0.149440
0.170000   0.169180
0.180000   0.188860
0.210000   0.208460
0.230000   0.227980

 N Natural spline
P Non-Periodic spline
E Extrapolated spline
C End point Curvature adjusted spline
Enter the choice : p
 Enter the values of y[0] and y[n]
0.18 0.21

 The cubic splines are:
p0(x)=1992.6906(x- 0.1500)^3-98.4818(x- 0.1500)^2+ 2.1596(x-
0.1500)+0.1494400
     in [ 0.1500, 0.1700]
p1(x)=1992.6904(x- 0.1700)^3+21.0797(x- 0.1700)^2+ 1.5579(x-
0.1700)+0.1691800
     in [ 0.1700, 0.1800]
p2(x)=-745.9187(x- 0.1800)^3+80.8604(x- 0.1800)^2- 1.1012(x-
0.1800)+0.1888600
     in [ 0.1800, 0.2100]
p3(x)=-745.9187(x- 0.2100)^3+13.7277(x- 0.2100)^2+ 0.9998(x-
0.2100)+0.2084600
     in [ 0.2100, 0.2300]
The value of y at x=0.220000 is 0.219085
```

Integration

5.1 Write a program in C to find the value of integration $\int_0^1 (x^2+1)dx$ by Gauss-Legendre quadrature formula for 6 points.

Program:

```
#include<stdio.h>
#include<conio.h>
float f(float x){
	return (x*x+1);
}
int main(){
	float x[10],w[10],a,b,p,q,result;
	int i;
	printf("Enter the values of a and b \n");
	scanf("%f%f",&a,&b);
	x[1]=0.23861919;
	x[2]=-x[1];
	x[3]=0.66120939;
	x[4]=-x[3];
	x[5]=0.93246951;
	x[6]=-x[5];
	w[1]=w[2]=0.46791393;
	w[3]=w[4]=0.36076157;
	w[5]=w[6]=0.17132449;
	p=(a+b)/2;
	q=(b-a)/2;
	result=0;
	for(i=1;i<=6;i++){
		result=result+w[i]*f(p+q*x[i]);
	}
	result=result*q;
	printf("The value of the integration is %f ",result);
	getch();
	return 0;
}
```

Input and Output Section:

```
Enter the values of a and b
0 1
The value of the integration is 1.333333
```

5.2 Write a program in C to find the value of integration $\int_0^1 \left(\frac{1}{1+x^2}\right) dx$ by Monte-Carlo technique.

Program:

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
float g(float x){
        return (1/(1+x*x));
}
int main(){
        float x,y,Integration,sum=0.0,a,b;
        int i,N;
        srand(100);
        printf("Enter the sample size: \n");
        scanf("%d",&N);
        printf("Enter the value of a and b :\n");
        scanf("%f%f",&a,&b);
        for(i=0;i<N;i++){
                y=(float)rand()/RAND_MAX;
                x=a+(b-a)*y;
                sum=sum+g(x);
        }
        Integration=sum*(b-a)/N;
        printf("N    Integration \n");
        printf("%d      %f",N,Integration);
        return 0;
}
```

Input and Output Section:

```
Enter the sample size:
500
Enter the value of a and b :
0 1
N    Integration
500      0.808336
Enter the sample size:
1000
Enter the value of a and b :
0 1
N    Integration
1000      0.797864
Enter the sample size:
1500
Enter the value of a and b :
0 1
N    Integration
1500      0.788843
Enter the sample size:
3000
Enter the value of a and b :
0 1
N    Integration
3000      0.788619
Enter the sample size:
10000
Enter the value of a and b :
0 1
N    Integration
10000      0.786627
Enter the sample size:
15000
Enter the value of a and b :
0 1
N    Integration
15000      0.786039
```

5.3 Write a program in C to find the value of the double integration of the function $\boldsymbol{F(x,y)=\int_0^1\int_0^1\left(\frac{1}{(1+x^2)(1+y^2)}\right)dxdy}$ ***by trapezoidal rule.***

Program:

```
#include<stdio.h>
#include<conio.h>
int m,n;
float f[20][20];
float sum(int i){
        float t=0;
        int j;
        for(j=1;j<m;j++){
                t=t+f[i][j];
        }
        return (f[i][0]+f[i][m]+2*t);
}
float F(float x,float y){
        return (1/((1+x*x)*(1+y*y)));
        //return (1+y*y)/(1+x*x);
}
int main(){
        int i,j;
        float a,b,c,d,h,k,x,y,result;
        printf("Enter the limits of x and y {a b : c d} \n");
        scanf("%f%f%f%f",&a,&b,&c,&d);
        printf("Enter the number of subdivisions n,m of x,y \n");
        scanf("%d%d",&m,&n);
        h=(b-a)/n;
        k=(d-c)/m;
        x=a;
        result=0;
        for(i=0;i<n;i++){
                y=c;
                for(j=0;j<m;j++){
                        f[i][j]=F(x,y);
                        y=y+k;
                }
```

```
            x=x+h;
        }
        for(i=0;i<n;i++){
            result=result+sum(i);
        }
        result=(sum(0)+2*result+sum(n))*h*k/4;
        printf("The value of the integration is %8.5f",result);
        return 0;
}
```

Input and Output Section:

```
Enter the limits of x and y {a b : c d}
0 1 0 1
Enter the number of subdivisions n,m of x,y
10 10
The value of the integration is  0.65357
```

Differentiation

6.1 Write a program in C to compute y (0.03) where $\frac{dy}{dx} = 2x^2 + 3y$***, y (0) =1 by Euler's method.***

Solution:

Here initial value x_0=0, y_0=1 and last value of x=0.03

Let h=0.*03* $n = \frac{x-x0}{h} = \frac{0.03-0}{0.01} = 3$

Program:

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
float f(float x,float y){
return (2*x*x+3*y);
}
int main(){
        float x0,y0,x,h,n;
        int i;
        clrscr();
        printf("Enter the initial value of x0 y0 and h \n");
        scanf("%f%f%f",&x0,&y0,&h);
        printf("Enter the last value of x \n");
        scanf("%f",&x);
        n=(x-x0)/h;
        for(i=1;i<=n;i++){
        y0=y0+h*f(x0,y0);
        x0=x0+h;
        }
        printf("The result y(%.2f)=%.3f ",x0,y0);
getch();
return 0;
}
```

Input and Output Section:

```
Enter the initial value of x0 y0 and h
0 1 0.01
Enter the last value of x
0.03
The result y (0.03) =1.093
```

6.2 Write a program in C to determine the value of y when x=0.2 given that y (0) =1 and $\frac{dy}{dx} = x^2 - y$ by Modifier Euler's method.

Solution:

Let h=0.1, x_0=0, y_0=1, x_n=0.2

Program:

```
#include<stdio.h>
#include<math.h>
float f(float x,float y){
        return (x*x-y);
}
int main(){
        float x0,y0,xn,h,x,y,eps=0.0001,yc,yp,f1;
        printf("Enter the initial value of x0 and final value of xn values of x: \n");
        scanf("%f%f",&x0,&xn);
        printf("Enter the initial value of y : \n");
        scanf("%f",&y0);
        printf("Enter the step length h: \n");
        scanf("%f",&h);
        printf("x_value   y_value \n");
        y=y0;
        for(x=x0;x<xn;x=x+h){
        f1=f(x,y);
        yc=y+h*f1;
        do{
                yp=yc;
                yc=y+h*(f1+f(x+h,yp))/2;
        }while(fabs(yp-yc)>eps);
        y=yc;
        printf("%f %f \n",x+h,y);
        }
        return 0;
}
```

Input and Output Section:

```
Enter the initial value of x0 and final value of xn values of x:
0 0.2
Enter the initial value of y :
1
Enter the step length h:
0.1
x_value   y_value
0.100000 0.905239
0.200000 0.821407
```

6.3 Write a program in C to find y(0.4) by solving the differential equation $\frac{dy}{dx} = x^2 - y^2$, y(0)=1 by fourth order Runge-Kutta method using step length 0.1.

Solution:

Here initial value of x_0=0, y_0=1

Last value of x=0.4 and h=0.1 then, n=$\frac{x-x0}{h} = \frac{0.4-0}{0.1} = 4$

Program:

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
float f(float x, float y){
        return (x*x-y*y);
}
int main(){
float x0,y0,x,h,k1,k2,k3,k4,n;
int i;
        //clrscr();
        printf("Enter the initial value of x0, y0 and h \n");
        scanf("%f%f%f",&x0,&y0,&h);
printf("Enter the last value of x \n");
         scanf("%f",&x);
n=(x-x0)/h;
for(i=1;i<=n;i++){
                k1=h*f(x0,y0);
                k2=h*f(x0+h/2,y0+k1/2);
```

```
                    k3=h*f(x0+h/2,y0+k2/2);
                    k4=h*f(x0+h,y0+k3);
                    y0=y0+(k1+2*k2+2*k3+k4)/6;
                    x0=x0+h;
                }
            printf("The result y(%.1f)=%f",x0,y0);
            getch();
        return 0;
        }
```

Input and Output Section:

```
Enter the initial value of x0, y0 and h
0 1 0.1
Enter the last value of x
0.4
The result y(0.4)=0.732728
```

6.4 Write a program in C to find y (0.4) by solving the differential equation $\frac{dy}{dx} = xy + y^2$, y (0) =1 by Milne Predictor Corrector method using step length 0.05.

Solution:

Here initial value of $x_0=0$, $y_0=1$

Last value of x=0.4 and h=0.05

Program:

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
float f(float x,float y){
        return (x*y+y*y);
}
float rk4(float x,float y,float h){
        float k1,k2,k3,k4;
        k1=h*f(x,y);
        k2=h*f(x+h/2,y+k1/2);
        k3=h*f(x+h/2,y+k2/2);
        k4=h*f(x+h,y+k3);
        y=y+(k1+2*(k2+k3)+k4)/6;
        //x=x+h;
        return (y);
}

int main(){
        float x0,y0,xn,h,y1,y2,y3,yc,yp;
        float x1,x2,x3,x4,f0,f1,f2,f3,yold,eps=0.01;
        printf("Enter the initail values of x and y: \n");
        scanf("%f%f",&x0,&y0);
        printf("Enter the last value of x: \n");
        scanf("%f",&xn);
        printf("Enter the step length of h: \n");
        scanf("%f",&h);
        printf("x_value  y_value: \n");
        //initial valuse of y are computed using RungeKutta method
        x1=x0+h;
        x2=x1+h;
        x3=x2+h;
        y1=rk4(x0,y0,h);
        y2=rk4(x1,y1,h);
        y3=rk4(x2,y2,h);
        f1=f(x1,y1);
        f2=f(x2,y2);
        f3=f(x3,y3);
        for(x4=x3+h;x4<=xn;x4=x4+h){
                yp=y0+4*h*(2*f1-f2+2*f3)/3.;
                yold=yp;
                //yc=yp;
                do{
                        yold=yc;
                        yc=y2+h*(f2+4*f3+f(x4,yold))/3.;
                }while(fabs(yc-yold)>eps);

                printf("%.5f %.5f \n",x4,yc);

                y0=y1;
                y1=y2;
                y2=y3;
                y3=yc;
                f1=f2;
                f2=f3;
                f3=f(x4,yc);
        }
        return 0;
}
```

Input and Output Section:

```
Enter the initail values of x and y:
0 1
Enter the last value of x:
0.4
Enter the step length of h:
0.05
x_value  y_value:
0.20000 1.27733
0.25000 1.38024
0.30000 1.50365
0.35000 1.65324
```

Eigenvalue and Eigenvectors of a Matrix

7.1 Write a program in C to find the largest eigen value and the corresponding Eigen vector of the following matrix using Power method:

$$\begin{bmatrix} 1 & 3 & 2 \\ -1 & 0 & 2 \\ 2 & 4 & 5 \end{bmatrix}$$

Program:

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
int main(){
        int n,i,j,flag;
        float a[10][10],x0[10],x1[10],y[10],lamda,eps=0.0001;
        printf("Enter the size of the matrix \n");
        scanf("%d",&n);
        printf("Enter the elements row wise \n");
        for(i=1;i<=n;i++){
                for(j=1;j<=n;j++){
                        scanf("%f",&a[i][j]);
                }
        }
        /*Printing of A*/
        printf("The given matrix is \n");
        for(i=1;i<=n;i++){
                for(j=1;j<=n;j++){
                        printf("%f",a[i][j]);
                }
                printf("\n");
        }
        printf("\n");
        for(i=1;i<=n;i++){
                /*Initialization*/
                x0[i]=1;
                x1[1]=1;
        }
        do{
                flag=0;
                /*reset x0*/
                for(i=1;i<=n;i++){
                        x0[i]=x1[i];
                }
                /*product of A and x0*/
                for(i=1;i<=n;i++){
                        y[i]=0;
                        for(j=1;j<=n;j++){
                                y[i]=y[i]+a[i][j]*x0[j];
                        }
                }
                /*finding maximum among y[i]*/
                lamda=y[1];
                for(i=2;i<=n;i++){
                        if(lamda<y[i]){
                                lamda=y[i];
                        }
                }

                for(i=1;i<=n;i++){
                        x1[i]=y[i]/lamda;
                }
                for(i=1;i<=n;i++){
                        if(fabs(x0[i]-x1[i])>eps)
                        flag=1;
                }
        }while(flag==1);
        printf("The largest eigenvalue is %8.5f \n",lamda);
        printf("The corresponding eigenvector is \n");
        for(i=1;i<=n;i++){
                printf("%8.5f ",x1[i]);
        }
        getch();
        return 0;
}
```

Input and Output Section:

```
Enter the size of the matrix
3
Enter the elements row wise
```

```
1 3 2
-1 0 2
3 4 5
The given matrix is
1.0000003.0000002.000000
-1.0000000.0000002.000000
3.0000004.0000005.000000

The largest eigenvalue is  7.16775
The corresponding eigenvector is
 0.43073  0.21893  1.00000
```

7.2 Write a program in C to find all the eigen values and the corresponding Eigen vectors of a real symmetric matrix

$$A=\begin{bmatrix}1 & 2 & 3 & 4\\ 2 & -3 & 3 & 4\\ 3 & 3 & 4 & 5\\ 4 & 4 & 5 & 0\end{bmatrix}$$

Using Jacobi's method.

Program:

```
#include<stdio.h>
#include<math.h>
#include<conio.h>
int main(){
	int n,i,j,p,q,flag;
	float a[10][10],d[10][10],s[10][10],s1[10][10],s1t[10][10];
	float temp[10][10],theta,zero=0.0001,pi=3.141592654,max;
	printf("Enter the size of a matrix: \n");
	scanf("%d",&n);
	printf("Enter the elements row wise: \n");
	for(i=1;i<=n;i++){
		for(j=1;j<=n;j++){
			scanf("%f",&a[i][j]);
		}
	}
	printf("The given matrix is \n");
	//Printing of A
	for(i=1;i<=n;i++){
		for(j=1;j<=n;j++){
			printf("%3.5f ",a[i][j]);
		}
			printf("\n");
	}
	printf("\n");
	//Initialize of D and S
	for(i=1;i<=n;i++){
		for(j=1;j<=n;j++){
			d[i][j]=a[i][j];
			s[i][j]=0;
		}
	}
	for(i=1;i<=n;i++){
		s[i][i]=1;
	}
	do{
		flag=0;
		//Find largest off-diagonal elements
		i=1;
		j=2;
		max=fabs(d[1][2]);
		for(p=1;p<=n;p++){
			for(q=1;q<=n;q++){
				if(p!=q){
					if(max<fabs(d[p][q])){
						max=fabs(d[p][q]);
						i=p;
						j=q;
					}
				}
			}
	}
		if(d[i][i]==d[j][j]){
			if(d[i][j]>0){
				theta=pi/4;
			}
			else{
				theta=-pi/4;
			}
```

```
}
else{
        theta=0.5*atan(2*d[i][j])/(d[i][i]-d[j][j]);
}
//Construction of the matrix s1 and s1t
for(p=1;p<=n;p++){
        for(q=1;q<=n;q++){
                s1[p][q]=0;
                s1t[p][q]=0;
        }
}
        for(p=1;p<=n;p++){
                s1[p][p]=1;
                s1t[p][p]=1;
        }
        s1[i][i]=cos(theta);
        s1[j][j]=s1[i][i];
        s1[j][i]=sin(theta);
        s1[i][j]=-s1[j][i];
        s1t[i][i]=s1[i][i];
        s1t[j][j]=s1[j][j];
        s1t[i][j]=s1[j][i];
        s1t[j][i]=s1[i][j];
        //Product of S1t and D
        for(i=1;i<=n;i++){
                for(j=1;j<=n;j++){
                        temp[i][j]=0;
                        for(p=1;p<=n;p++){
                                temp[i][j]=temp[i][j]+s1t[i][p]*d[p][j];
                        }
                }
        }
        //Product of temp and S1 that is D=S1T*S1
        for(i=1;i<=n;i++){
        for(j=1;j<=n;j++){
                d[i][j]=0;
                for(p=1;p<=n;p++){
                        d[i][j]=d[i][j]+temp[i][p]*s1[p][j];
                }
        }
```

```
                }
        //Product of S and s1 that is S=S*S1
        for(i=1;i<=n;i++){
                for(j=1;j<=n;j++){
                        temp[i][j]=0;
                        for(p=1;p<=n;p++){
                                temp[i][j]=temp[i][j]+s[i][p]*s1[p][j];
                        }
                }
        }
        for(i=1;i<=n;i++){
                for(j=1;j<=n;j++){
                        s[i][j]=temp[i][j];
                }
        }
        //D is Diagonal
        for(i=1;i<=n;i++){
                for(j=1;j<=n;j++){
                        if(i!=j){
                                if(fabs(d[i][j]>zero)){
                                        flag=1;
                                        }
                                }
                        }
                }
        }while(flag==1);
        printf("The Eign values are: \n");
        for(i=1;i<=n;i++){
                printf("%3.5f ",d[i][i]);
        }
        printf("\n The corresponding Eign vectors are: \n");
        for(j=1;j<=n;j++){
                printf("(");
                for(i=1;i<n;i++){
                        printf("%3.5f ",s[i][j]);
                }
                printf("%3.5f ) \n",s[n][j]);
        }
        return 0;
}
```

Input and Output Section:

```
Enter the size of a matrix:
4
Enter the elements row wise:
1 2 3 4
2 -3 3 4
3 3 4 5
4 4 5 0
The given matrix is
1.00000 2.00000 3.00000 4.00000
2.00000 -3.00000 3.00000 4.00000
3.00000 3.00000 4.00000 5.00000
4.00000 4.00000 5.00000 0.00000

The Eign values are:
-0.73369 -5.88321 11.78253 -3.16564
 The corresponding Eign vectors are:
(0.74263 0.04637 -0.65235 0.14420 )
(0.13467 0.74459 0.06236 -0.65081 )
(0.43845 0.33396 0.64096 0.53422 )
(-0.48798 0.57611 -0.39965 0.51986 )
```