Assignment 1:

Write a program in C to calculate f (0.5) by Newton’s Forward Interpolation formula using the following data:

x	0	1	2	3
$f(x)$	1	0	1	10

Answer: 0.625

Algorithm for Newton’s Forward Interpolation formula:

```
1 . Start
2. Read n
3. For i=0 to (n-1) do
Read xi
Next i
4. For i=0 to (n-1) do
Read yi,0
Next i
5. Read X
6. For j=1 to (n-1) do
        For i=j to (n-1) do
yi,j =yi,j-1 - yi-1,j-1
Next i
Next j
7. For i=0 to (n-1) do
    For j=0 to i do
 Print yi,j
Next j
Next i
8. h = x1-x0
9. u = (X-x0)/h
10. term =1.0
11. Sum = y0,0
12. For i=1 to (n-1) do
       term = term * ( u-i+1)/i
Sum = sum + term * yi,i
Next i
13. Print Sum
14. Stop
```

Program for Newton’s Forward Interpolation formula

```c
/*program for newton's forward interpolation*/
#include<stdio.h>
#include<stdlib.h>
int main()
{           float x[20],y[20][20],s,t=1.0,X,h,u;
int i,j,n;
printf("enter n:");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("enter x%d:",i+1);
scanf("%f",&x[i]);
printf("enter y%d:",i+1);
scanf("%f",&y[i][0]);
}
printf("enter the point of interpolation:");
scanf("%f",&X);
for(j=1;j<n;j++)
{
for(i=j;i<n;i++)
y[i][j]=y[i][j-1]-y[i-1][j-1];
}
printf("difference table\n\n");
for(i=0;i<n;i++)
{
for(j=0;j<=i;j++)
printf("%7.3f",y[i][j]);
printf("\n");
}
h=x[1]-x[0];
u=(X-x[0])/h;
s=y[0][0];
for(i=1;i<n;i++)
{
t=t*(u-i+1)/i;
s=s+t*y[i][i];
}
printf("\nvalue of the function:");
printf("at x=%f is %0.3f",X,s);
return 0;}
```

Output:

```
enter n:4
enter x1:0
enter y1:1
enter x2:1
enter y2:0
enter x3:2
enter y3:1
enter x4:3
enter y4:10
enter the point of interpolation:0.5
difference table

  1.000
  0.000 -1.000
  1.000  1.000  2.000
 10.000  9.000  8.000  6.000

value of the function:at x=0.500000 is
0.625
--------------------------------
Process exited after 64.99 seconds
with return value 0
Press any key to continue . . .
```

Assignment 2:

Write a program in C to calculate *f* (2.5) by Newton’s Backward Interpolation formula using the following data:

x	0	1	2	3
$f(x)$	1	0	1	10

Answer: 4.125

Algorithm for by Newton’s Backward Interpolation formula:

```
1. Start
2. Read n
3. For i=0 to (n-1) do
    Read xi
   Next i
4. For i= 0 to (n-1) do
    Read yi,0
   Next i
5. h=x1-x0
6. For i=1 to (n-2) do
    If x(i+1)-x(i)! ≠ h
    Then flag =0
    End of if
   End of flag
7. If flag ==1
    Print (" formula is applicable")
8. Read X
9. For j=1 to (n-1) do
    For i=j to ( n-1) do
        yi,j= yi,j-1 - yi-1,j-1
    Next i
   Next j
10. For i=0 to (n-1) do
    For j=0 to 1 do
        Print yi,j
    Next j
   Next i
11. u = (X-x[n-1])/h
12. Term = 1.0
13. Sum= y n-1,0
14. For i=1 to ( n-1) do
    Term = term * (u+i-1) /i
    Sum = sum +t* y n-1,i
   Next i
15. Print Sum
16. Stop
```

Program for Newton’s Backward Interpolation formula:

```c
/*program for newton's backward interpolation*/
#include<stdio.h>
#include<stdlib.h>
int main()
{
float x[20],y[20][20],s,t=1.0,X,h,u;
int i,j,n,flag=1;
printf("enter n:");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("enter x%d:",i+1);
scanf("%f",&x[i]);
printf("enter y%d:",i+1);
scanf("%f",&y[i][0]);
}
h=x[1]-x[0];
for(i=1;i<n-1;i++)
{
if((x[i+1]-x[i])!=h)
{
flag=0;
break;
}
}
if(flag==1)
{
printf("newton's backward interpolation formula is applicable\n");
printf("enter the point of interpolation:");
scanf("%f",&X);
for(j=1;j<n;j++)
{
for(i=j;i<n;i++)
y[i][j]=y[i][j-1]-y[i-1][j-1];
}
printf("difference table\n\n");
for(i=0;i<n;i++)
{
for(j=0;j<=1;j++)
printf("%7.3f",y[i][j]);
printf("\n");
}
u=(X-x[n-1])/h;
s=y[n-1][0];
for(i=1;i<n;i++)
{
t=t*(u+i-1)/i;
s=s+t*y[n-1][i];
}
printf("\nvalue of the function:");
printf("at x=%f is %0.3f",X,s);
}
else
printf("newton's backward interpolation is not applicable");
return 0;
}
```

Output

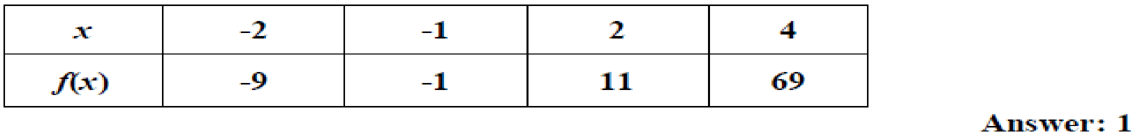
```
enter n:4
enter x1:0
enter y1:1
enter x2:1
enter y2:0
enter x3:2
enter y3:1
enter x4:3
enter y4:10
newton's backward interpolation formula is applicable
enter the point of interpolation:2.5
difference table

  1.000  0.000
  0.000 -1.000
  1.000  1.000
 10.000  9.000

value of the function:at x=2.500000 is 4.125
--------------------------------
Process exited after 63.2 seconds with return value 0
Press any key to continue . . .
```

Assignment 3:

Write a program in C to find *f* (*x*) for x=0 by Lagrange's Interpolation formula using the following data:



x	-2	-1	2	4
f(*x*)	-9	-1	11	69

Answer: 1

Algorithm for by Lagrange's Interpolation formula:

```
1. Start
2. For i=0 to (n-1) do
Read xi,hi
Next i
3. Read x
4. Set s=0.0
5. For i=0 to (n-1) do
     Set p=1.0
    for j=0 to (n-1) do
If i≠j, compute p=p*(x-xj)/(xi-xj)
Next j
s=s+p*yi
Next i
6. Print s
7. Stop
```

Program for Lagrange’s Interpolation formula:

```c
/*program for lagrange's interpolation*/
#include<stdio.h>
#include<math.h>
int main()
{
float *x,*y,X,p,s=0.0;
int i,j,n;
printf("enter n:");
x=(float *)malloc(n*sizeof(float));
y=(float *)malloc(n*sizeof(float));
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("enter x%d:",i+1);
scanf("%f",&x[i]);
printf("enter y%d:",i+1);
scanf("%f",&y[i]);
}
printf("enter the point of interpolation:");
scanf("%f",&X);
for(i=0;i<n;i++)
{
p=1.0;
for(j=0;j<n;j++)
{
if(i!=j)
p=p*(X-x[j])/(x[i]-x[j]);
}
s=s+p*y[i];
}
printf("value of interpolation:");
printf("at %f is:%0.3f",X,s);
return 0;
}
```

Output:

```
enter n:4
enter x1:-2
enter y1:-9
enter x2:-1
enter y2:-1
enter x3:2
enter y3:11
enter x4:4
enter y4:69
enter the point of interpolation:0
value of interpolation:at 0.000000
is:1.000
```

Assignment 4:

Write a program in C by applying Trapezoidal Rule to find the value of the following definite integral:

$\int_0^1 \frac{dx}{1+x^2}$, correct the result up to 3 decimal places.

Answer: 0.785

Algorithm for Trapezoidal Rule:

```
1. Start
2. Read a,b
3. Read n
4. h=(b-a)/n
5. Set s = 0.0
6. Set i=0
7. s=s+f(a+(i+0)*h)+f(a+(i+1)*h
8. i=i+1
9. If i<n , go to step 7 ,else go to next step
10. s=s*(h/2)
11. Print s
12. Stop
```

Program for Trapezoidal Rule:

```c
/*TRAPEZOIDAL RULE*/
#include<stdio.h>
#include<math.h>
 /* Define the function to be integrated here: */
float f(float x)
{
  return 1/(1+x*x);
}
 /*Program begins*/
main()
{
  int n,i;
  float a,b,h,x,sum=0,integral;
  /*Ask the user for necessary input */
  printf("\nEnter the initial limit: ");
  scanf("%f",&a);
  printf("\nEnter the final limit: ");
  scanf("%f",&b);
  printf("\nEnter the no. of sub-intervals: ");
  scanf("%d",&n);
  h=fabs(b-a)/n;
  for(i=1;i<n;i++)
  {
    x=a+i*h;
    sum=sum+f(x);
  }
  integral=(h/2)*(f(a)+f(b)+2*sum);
  /*Print the answer */
  printf("\nThe integral is: %f\n",integral);
}
```

Output

```
Enter the initial limit:
 0
Enter the final limit:
 1
Enter the no. of sub-intervals:
 10

The integral is: 0.784981

--------------------------------
Process exited after 9.936
seconds with return value 0
Press any key to continue . . .
```

Assignment 5:

Write a program in C by applying Simpson’s 1/3 Rule to find the value of the following definite integral:

$\int_0^1 \frac{xdx}{1+x}$, correct the result up to 3 decimal places.

Answer: 0.307

Algorithm for SIMPSON’S 1/3 RULE:

```
1. Start
2. Read a,b
3. Read n
4. If n%2 =0 , go to next step ,else go to step 13
5. h =(b-a)/n
6.set s=0.0
7. Set i=0
8. S=s+f(a+(i+0)*h)+4*f(a+(i+1)*h)+f(a+(i+2)*h)
9. i=i+2
10. If i<n, go to step 8 ,else go to next step
11.s=s*(h/3)
12. Print s ,go to step 14
13. Print " Simpson's 1/3 rule is not applicable"
14. Stop
```

Program for SIMPSON'S 1/3 RULE:

```
/*SIMPSON'S 1/3 RULE*/
#include<stdio.h>
#include<math.h>
float f(float x)
{
return x*x;
}
main()
{
int n,i;
float a,b,h,x,sum=0,integral;
printf("\nEnter the initial limit: ");
scanf("%f ",&a);
printf("\nEnter the final limit: ");
scanf("%f ",&b);
printf("\nEnter the no. of sub-intervals(EVEN): ");
scanf("%d",&n);
h=fabs(b-a)/n;
if(n%2==0)
{
for(i=1;i<n;i++)
{
x=a+i*h;
if(i%2==0)
{
sum=sum+2*f(x);
}
else
{
sum=sum+4*f(x);
}
}
integral=(h/3)*(f(a)+f(b)+sum);
printf("\nThe integral is: %f \n",integral);
}
else
printf(" error");
}
```

Output:

```
Enter the initial limit:
0
Enter the final limit:
1
Enter the no. of sub-intervals(EVEN):
6

The integral is: 0.306830

--------------------------------
Process exited after 5.286 seconds with return value 0
Press any key to continue . . .
```

Assignment 6:

Write a program in C by applying Weddle’s Rule to find the value of the following definite integral:
$\int_0^{-1} xe^x$, correct the result up to 3 decimal places.

Answer: -0.2616

Algorithm for Weddle’s Rule:

```
1.start
2. Read a,b
3.Read n
4. If n%6=0 , go to next step , else go to
5. h =(b-a)/n
6. Set s=0.0
7. Set i=0
8. s= s+f(a+(i+0)*h)+5*f(a+(i+1)*h)+f(a+(i+2)*h)+6*f(a+(i+3)*h)+f(a+(i+4)*h)+5*f(a+(i+5)*h)+f(a+(i+6)*h)
9. i=i+6
10. If i<n ,go to step 8, else go to next step
11. s=s*(3h/10)
12. Print s , go to step 14
13. Print " weddle's rule is not applicable"
14. Stop
```

Program for Weddle’s Rule:

```c
/* Weddle’s Rule */
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
float f(float x)
{
return (x*exp(x));
}
int main()
{
float a,b,h,s=0;
int n,i;
printf("enter lower limit:");
scanf("%f",&a);
printf("enter upper limit:");
scanf("%f",&b);
printf("enter subintervals:");
scanf("%d",&n);
if(n%6==0)
{
h=(b-a)/n;
for(i=0;i<n;i=i+6)
s=s+f(a+(i+0)*h)+5*f(a+(i+1)*h)+f(a+(i+2)*h)+6*f(a+(i+3)*h)+f(a+(i+4)*h)+5*f(a+(i+5)*h)+f(a+(i+6)*h);
s=s*(3*h/10);
printf("the value is:%f",s);
}
else
printf(" rule is not applicable");
}
```

Output:

```
enter lower limit:0
enter upper limit:-1
enter subintervals:12
the value is:0.264241
--------------------------------
Process exited after 6.345
seconds with return value 21
Press any key to continue . . .
```

Assignment 7:

Write a program in C by applying Regula-Falsi method to find the value of the following algebraic equation:
$X^3 + 2x +1=0$, correct the result up to 3 decimal places.

Answer: -1.168

Algorithm for Regula-Falsi method:

```
f(x)=0,a,b,c
1. Start
2. Read a,b
3. If f(a)*f(b)<0 ,go to next step ,else go to step 2
4. Read e
5. x = (a *f(b)-b*f(a)/f(b)-f(a))
6. If f(a)*f(b)<0 ,go to next step ,else go to step 8
7. b=x go to step 9
8. a=x go to next step
9. If |a-b|<e , go to step 10 , go to step 5
10. Print x
11. Stop
```

Program for Regula-Falsi method:

```c
/*Regula Falsi mthod*/
#include <stdio.h>
#include <math.h>
float f(float x)
{
return (x*x*x-2*x+1);
}
int main()
{
float a,b,e,x;
while(1)
{
printf("enter the value of a & b:");
scanf("%f%f",&a,&b);
if(f(a)*f(b)<0.0)
break;
printf("enter a new intervals again");
}
printf("enter error value:");
scanf("%f",&e);
do
{
x=(a*f(b)-b*f(a))/(f(b)-f(a));
if (f(x)*f(a)<0.0)
b=x;
else
a=x;
}
while (fabs(a-b)>0.0);
{
printf("one real root is :%0.3f",x);
}
}
```

Output:

```
enter the value of a & b:
-1
-2
enter error value:0.001
one real root is :-1.618
--------------------------------
Process exited after 5.896
seconds with return value 0
Press any key to continue . . .
```

Assignment 8:

Write a program in C by applying Newton-Raphson method to find the value of the following algebraic equation:

$x^3 + 2x + 1 = 0$, correct the result up to 3 decimal places.

Answer: -1.168

Algorithm for Newton-Raphson method:

```
f(x) =0, x0,e
1. start
2. Read x0
3. Read e
4. x = x0
5. x0= x0-f(x0)/F1(x0)
6. If |x-x0|<e ,go to step 7 ,else go to step 4
7. Print x
8. Stop
```

Program for Newton-Raphson method:

```c
/* Newton raphjson method */
#include<stdio.h>
#include<math.h>
float f(float z)
{
return (z*z*z-2*z+1);
}
float f1(float z)
{
return (3*z*z-2);
}
int main ()
{
float x0,x,e;
printf(" enter the initial guess of root : ");
scanf(" %f",&x0);
printf(" enter the error value:");
scanf("%f",&e);
do
{
x=x0;
x0=x0-f(x0)/f1(x0);
}
while (fabs(x-x0)>e);
printf("one real root of the equation is :%0.3f",x);
}
```

Output:

```
enter the initial guess of root
: -2
 enter the error value:0.001
one real root of the equation
is :-1.618
--------------------------------
Process exited after 9.981
seconds with return value 0
Press any key to continue . . .
```

Assignment 9:

Write a program in C by applying Euler’s method to find the solution of the following differential equation:

$\frac{dx}{dy} = \frac{x-y}{x+y}$, taking y=1 at x=0, find y at x=0.1, taking step length 0.02 and print the result correct up to 3 decimal places. **Answer: 1.093**

Algorithm for eular method:

```
1.start
2. Read x0,y0
3. Read xn
4. Read h
5. y0= y0+h*f(x0,y0)
6. x0=x0+h
7. If x0<xn ,go to step 5,go to next step
8. Print y0
9. Stop
```

Program for eular method:

```c
/*eular method*/
#include<stdio.h>
#include<math.h>
float f(float x,float y)
{
return (y-x)/(y+x);
}
int main()
{
float x0,y0,xn,h;
printf(" enter the value of x0,y0 & xn:\n");
scanf("%f%f%f",&x0,&y0,&xn);
printf("enter step length:\n");
scanf("%f",&h);
do
{
y0=y0+h*f(x0,y0);
x0=x0+h;
}
while(x0<xn);
printf("the value of y0 is:%f",y0);
}
```

Output:

```
enter the value of x0,y0 & xn:
0
1
0.1
enter step length:
.02
the value of y0 is:1.109478
--------------------------------
Process exited after 7.022 seconds with
return value 0
Press any key to continue . . .
```

Assignment 10:

Write a program in C by applying R-K method of order 4 to find the solution of the following differential equation: $\frac{dx}{dy} = \frac{x-y}{x+y}$, taking y=1 at x=0, find y at x=0.1,taking step length 0.02 and print the result correct up to 3 decimal places.

Answer: 1.092

Algorithm for R-K method of order 4 :

```
1.start
2.Read x0,y0,xn,h
3.k1=h*f( x0,y0)
4.k2=h*f( x0+h/2,y0+k1/2)
5.k3=h*f( x0+h/2,y0+k2/2)
6.k4=h*f( x0+h,y0+k3)
7.y0=y0+(k1+2*(k2+k3)+ke)/6
8.x0=x0+h
9.If x0<=xn,go to step 3 ,go to next step
10.print y0
11.stop
```

Program for R-K method of order 4 :

```c
/*program for rk method*/
#include<stdio.h>
#include<math.h>
float f(float,float);
int main()
{
float x0,y0,xn,k1,k2,k3,k4,h;
printf("enter x0:");
scanf("%f",&x0);
printf("enter y0:");
scanf("%f",&y0);
printf("enter xn:");
scanf("%f",&xn);
printf("enter step length:");
scanf("%f",&h);
do
{
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+k3)+k4)/6;
x0=x0+h;
}
while(x0<xn);
printf("the solution is:%0.3f",y0);
return 0;
}
float f(float x,float y)
{
return (y-x)/(y+x);
}
```

Output:

```
enter x0:0
enter y0:1
enter step length:0.02
enter xn:0.1
the solution is:1.109
```

Assignment 11:

Write a program in C using Gauss-Seidel iterative method to solve the following system of linear equations:

$3x + y + 5z = 13$

$5x - 2y + z = 4$

$x + 6y – 2z = -1$

Answer: x=0.55, y=0.47, z=2.18 Algorithm for Gauss-Seidel iterative method:

AX= b ,Ab

```
1.start
2. Read n
3. For i = 0 to (n-1) do
       For  j=0 to n do
           Read aij
       Next j
   Next i
4. Read count
5. Set all xi=0.0
6. If $|a_{ii}| > \sum_{i,j=0,\ i\neq j}^{n-1} |a_{ij}|$, go to next step, else go to step 12
7. Set k=1
8. For i=0 to (n-1) do
       Set xi = ai,n
       For j=0 to (n-1) do
           If i≠j ; compute xi= xi - aij*xj
       Next j
       xi= xi / aii
   Next i
9. If k<=count ,go to  step 8 ,else go to next step
10. For i=0 to (n-1) do
        Print xi
    Next i
11. Go to step 13
12. Print ( " gauss Seidel method cannot be applicable")
13. Stop
```

Program for Gauss-Seidel iterative method:

```
/* program for Gauss-Seidel method */
#include<stdio.h>
#include<math.h>
int main()
{
float a[10][10],A[10][10],s,max,x[10]={0.0};
int i,j,n,k,flag=0,pos,cnt;
printf("Enter n : ");
scanf("%d",&n);
printf("Enter Augmented matrix : \n");
for(i=0;i<n;i++)
{
for(j=0;j<n+1;j++)
{
scanf("%f",&a[i][j]);
}
}
for(i=0;i<n;i++)
{
max=a[i][0];
for(j=0;j<n;j++)
{
if(a[i][j]>=max)
{
max=a[i][j];
pos=j;
 }
}
s=0.0;
for(j=0;j<n;j++)
{
if(j!=pos)
{
s=s+fabs(a[i][j]);
}
}
if(fabs(a[i][pos])>s)
{
flag=1;
for(j=0;j<n+1;j++)
{
A[pos][j]=a[i][j];
}
}
else
{
flag=0;
break;
}
}
if(flag==1)
{
printf("System of equations are diagonally dominant\n");
printf("The augmented matrix after rearrangement of equations :\n");
for(i=0;i<n;i++)
{
for(j=0;j<n;j++)
{
printf("%7.3f",A[i][j]);
}
printf("\n");
}
printf("Enter no. iterations : ");
scanf("%d",&cnt);
k=0;
while(k<cnt)
{
for(i=0;i<n;i++)
{
x[i]=A[i][j];
for(j=0;j<n;j++)
{
if(i!=j)
{
x[i]=x[i]-A[i][j]*x[j];
}
}
x[i]=x[i]/A[i][i];
}
k++;
}
printf("Solutions : \n");
for(i=0;i<n;i++)
{
printf("x%d=%0.3f\n",i+1,x[i]);
}
}
else
{
printf("System of equations are not diagonally dominant\n");
printf("Gauss-Seidel Method cannot be applicable");
}
return 0;
}
```

- **Output:**

```
Enter n : 3
Enter Augmented matrix :
3 1 5 13
5 -2 1 4
1 6 -2 -1
System of equations are diagonally dominant
The augmented matrix after rearrangement of equations :
  5.000 -2.000  1.000
  1.000  6.000 -2.000
  3.000  1.000  5.000
Enter no. iterations : 25
Solutions :
x1=0.552
x2=0.467
x3=2.176
```

Assignment 12:

Write a program in C using Gauss-Elimination method to solve the following system of linear equations:

5*x1* – *x2* + *x3* = 10

2*x1* + 4*x2* = 12

x1 + *x2* + 5*x3* = -1

Answer: *x1*=2.556, *x2*=1.772, *x3*=-1.056

Algorithm for Gauss-Elimination method:

```
1.start
2. Read n
3. for i=0 to n do
    for j=0 to (n+1) do
        Read aij
    Next i
Next j
4. for k=0 to (n-1) do
    for i=(k+1) to n do
        compute ,r = a[i][j]/a[k][k]
        for j=0 to ( n+1) do
            a[i][j]=a[i][j]-r*a[k][j]
        Next j
Next k
5. for i= (n-1) to 0
    x[i]=a[i][n]
    for j=(i+1) to n
        x[i]=x[i]-a[i][j]*x[j]
    Next j
    x[i]=x[i]/a[i][j]
Next i
6. Print the result x[i]
7. Stop
```

***Program for Gauss-Elimination method*:**

```
program for gauss elimination method/
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
int main()
{
float **a,*x,r;
int n,i,j,k;
printf("enter n:");
scanf("%d",&n);
x=(float *)malloc(n*sizeof(float));
a=(float **)malloc(n*sizeof(float));
for(i=0;i<n+1;i++)
a[i]=(float *)malloc(n*sizeof(float));
printf("enter the augmented matrix:\n");
for(i=0;i<n;i++)
for(j=0;j<n+1;j++)
scanf("%f",&a[i][j]);
/elementary transformation/
for(k=0;k<n-1;k++)
{
for(i=k+1;i<n;i++)
{
r=a[i][k]/a[k][k];
for(j=0;j<n+1;j++)
a[i][j]=a[i][j]-r*a[k][j];
}
}
/printing of transformed augmented matrix/
for(i=0;i<n;i++)
{
for(j=0;j<n+1;j++)
printf("%7.3f",a[i][j]);
printf("\n");
}
/back substitution/
for(i=n-1;i>=0;i--)
{
x[i]=a[i][n];
for(j=i+1;j<n;j++)
x[i]=x[i]-a[i][j]*x[j];
x[i]=x[i]/a[i][i];
}
/printing of solutions/
printf("the solutions are:\n");
for(i=0;i<n;i++)
printf("x%d=%0.4f\n",i+1,x[i]);
return 0;
}
```

- **Output:**

```
enter n:3
enter the augmented matrix:
5 -1 1 10
2 4 0 12
1  1  5 -1
  5.000 -1.000  1.000 10.000
  0.000  4.400 -0.400  8.000
  0.000  0.000  4.909 -5.182
the solutions are:
x1=2.5556
x2=1.7222
x3=-1.0556
```