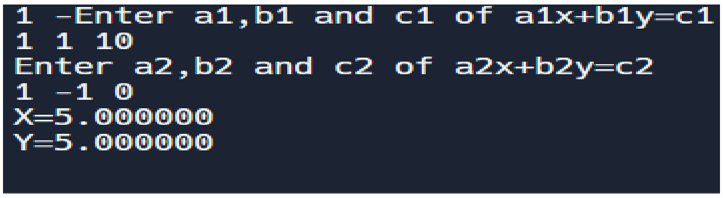
Simultaneous Equations:

Code:

```
#include<stdio.h>
#include<math.h>
int main()
{
    float a,b,c,p,q,r,x,y;
    printf("Enter a1,b1 and c1 of a1x+b1y=c1\n");
    scanf("%f%f%f",&a,&b,&c);
    printf("Enter a2,b2 and c2 of a2x+b2y=c2\n");
    scanf("%f%f%f",&p,&q,&r);
    q=q-p*b/a;
    r=r-p*c/a;
    if(q==0){
      printf("No Solutions\n");
    }
    else{
      y=r/q;
      x=(c-b*y)/a;
      printf("X=%f\nY=%f\n",x,y);
    }
    return 0;
}
```

Outputs:



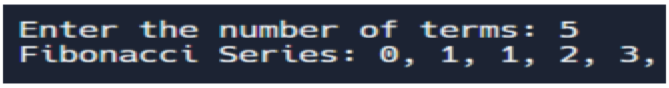
```
1 -Enter a1,b1 and c1 of a1x+b1y=c1
1 1 10
Enter a2,b2 and c2 of a2x+b2y=c2
1 -1 0
X=5.000000
Y=5.000000
```

Fibonacci Series:

Code:

```
#include <stdio.h>
int main() {
  int i, n;
  int t1 = 0, t2 = 1;
  int nextTerm = t1 + t2;
  printf("Enter the number of terms: ");
  scanf("%d", &n);
  printf("Fibonacci Series: %d, %d, ", t1, t2);
  for (i = 3; i <= n; ++i) {
    printf("%d, ", nextTerm);
    t1 = t2;
    t2 = nextTerm;
    nextTerm = t1 + t2;
  }
  return 0;
}
```

Output:



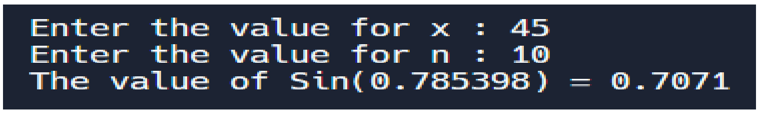
```
Enter the number of terms: 5
Fibonacci Series: 0, 1, 1, 2, 3,
```

<u>Sine Series:</u>

Code:

```
#include<stdio.h>
int main()
{
   int i, n;
   float x, sum, t;
   printf(" Enter the value for x : ");
   scanf("%f",&x);
   printf(" Enter the value for n : ");
   scanf("%d",&n);
   x=x*3.14159/180;
   t=x;
   sum=x;
   for(i=1;i<=n;i++)
   {
      t=(t*(-1)*x*x)/(2*i*(2*i+1));
      sum=sum+t;
   }
   printf(" The value of Sin(%f) = %.4f",x,sum);
   return 0;
}
```

Output:



```
 Enter the value for x : 45
 Enter the value for n : 10
 The value of Sin(0.785398) = 0.7071
```

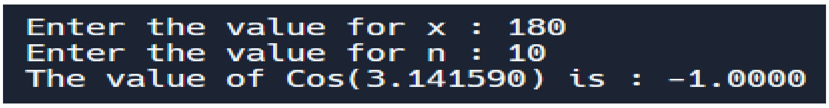
<u>Cosine Series:</u>

Code:

```
#include<stdio.h>
int main()
{
   int i, n;
   float x, sum=1, t=1;
   printf(" Enter the value for x : ");
   scanf("%f",&x);
   printf(" Enter the value for n : ");
   scanf("%d",&n);
   x=x*3.14159/180;
   for(i=1;i<=n;i++)
   {
      t=t*(-1)*x*x/(2*i*(2*i-1));
```

```
        sum=sum+t;
    }
    printf(" The value of Cos(%f) is : %.4f", x, sum);
    return 0;
}
```

Output:

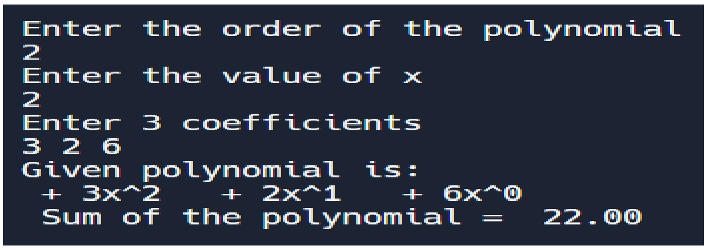


```
Enter the value for x : 180
Enter the value for n : 10
The value of Cos(3.141590) is : -1.0000
```

Polynomial Evaluation:

```
#include <stdio.h>
#include <stdlib.h>
#define MAXSIZE 100
void main()
{
    int array[MAXSIZE];
    int i, num, power;
    float x, polySum;
    printf("Enter the order of the polynomial \n");
    scanf("%d", &num);
    printf("Enter the value of x \n");
    scanf("%f", &x);
    printf("Enter %d coefficients \n", num + 1);
    for (i = 0; i <= num; i++)
    {
        scanf("%d", &array[i]);
    }
    polySum = array[0];
    for (i = 1; i <= num; i++)
    {
        polySum = polySum * x + array[i];
    }
    power = num;
    printf("Given polynomial is: \n");
    for (i = 0; i <= num; i++)
    {
        if (power < 0)
        {
            break;
        }
        if (array[i] > 0)
            printf(" + ");
        else if (array[i] < 0)
            printf(" - ");
        else
            printf(" ");
        printf("%dx^%d  ", abs(array[i]), power--);
    }
    printf("\n Sum of the polynomial = %6.2f \n", polySum);
}
```

Output:



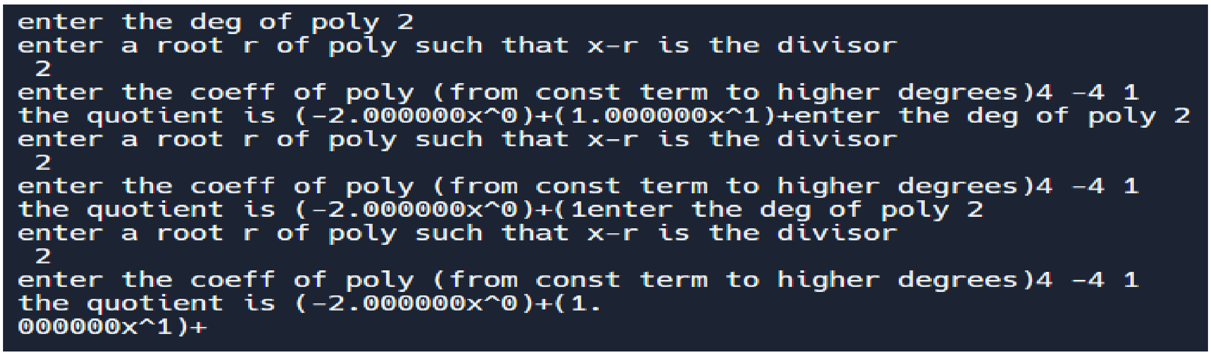
```
Enter the order of the polynomial
2
Enter the value of x
2
Enter 3 coefficients
3 2 6
Given polynomial is:
  + 3x^2    + 2x^1    + 6x^0
  Sum of the polynomial =   22.00
```

Polynomial Division:

Code:

```
#include<stdio.h>
#include<math.h>
int main()
{
    int n ;
    printf("enter the deg of poly ");
    scanf("%d",&n);
    float a[n+1],b[n],r;
    printf("enter a root r of poly such that x-r is the divisor\n ");
    scanf("%f",&r);
    printf("enter the coeff of poly (from const term to higher degrees)");
    for(int i=0;i<=n;i++){
        scanf("%f",&a[i]);
    }
    b[n-1]=a[n];
    for(int i=1;i<=n-1;i++){
        b[n-(i+1)]=a[n-1] + r*b[n-1];
    }
    printf("the quotient is ");
    for(int i=0;i<=n-1;i++){
        printf("(%fx^%d)+",b[i],i);
    }
    return 0;
}
```

Output:



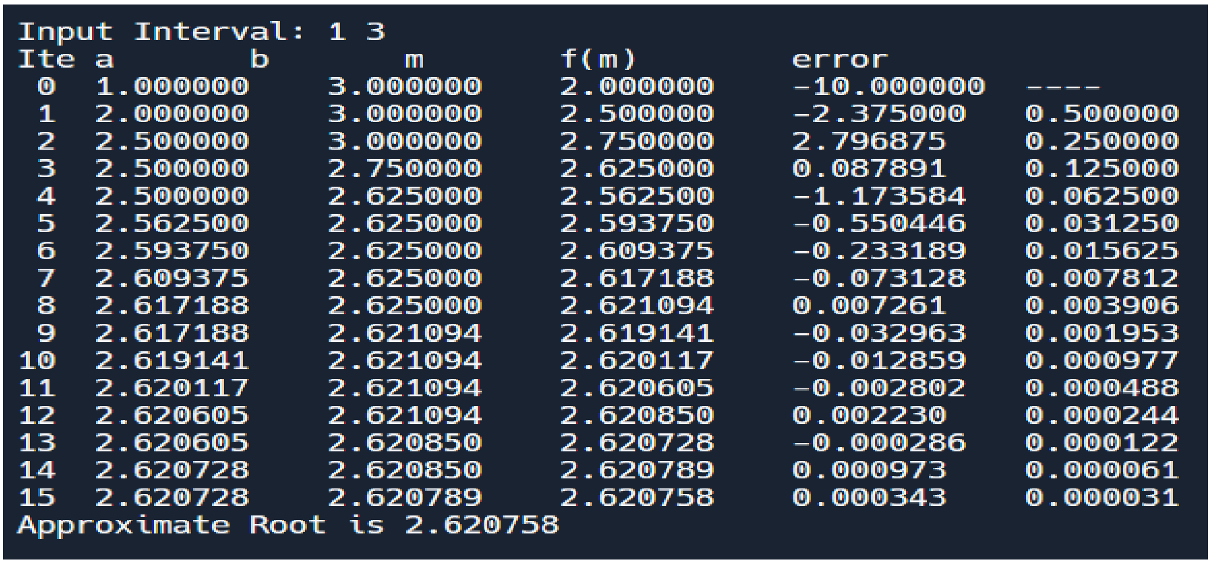
```
enter the deg of poly 2
enter a root r of poly such that x-r is the divisor
 2
enter the coeff of poly (from const term to higher degrees)4 -4 1
the quotient is (-2.000000x^0)+(1.000000x^1)+enter the deg of poly 2
enter a root r of poly such that x-r is the divisor
 2
enter the coeff of poly (from const term to higher degrees)4 -4 1
the quotient is (-2.000000x^0)+(1enter the deg of poly 2
enter a root r of poly such that x-r is the divisor
 2
enter the coeff of poly (from const term to higher degrees)4 -4 1
the quotient is (-2.000000x^0)+(1.
000000x^1)+
```

Bisection Method:

Code:

```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#define f(x) ((x * x * x) - 18)
int main() {
  float a = 0, b = 0, error = 0, m, mod;
  int i = 0;
  printf("Input Interval: ");
  scanf("%f %f", &a, &b);
  if ((f(a) * f(b)) > 0) {
    printf("Invalid Interval Exit!");
    exit(1);
  } else if (f(a) == 0 || f(b) == 0) {
    printf("Root is one of interval bounds. Root is %f\n", f(a) == 0 ? a : b);
    exit(0);
  }
  printf("Ite\ta\t\tb\t\tm\t\tf(m)\t\terror\n");
  do {
    mod = m;
    m = (a + b) / 2;
    printf("%2d\t%4.6f\t%4.6f\t%4.6f\t%4.6f\t", i++, a, b, m, f(m));
    if (f(m) == 0) {
      printf("Root is %4.6f\n", m);
    } else if ((f(a) * f(m)) < 0) {
      b = m;
    } else
      a = m;
    error = fabs(m - mod);
    if (i == 1) {
      printf("----\n");
    } else
      printf("%4.6f\n", error);
  } while (error > 0.00005);
  printf("Approximate Root is %4.6f", m);
  return 0;
}
```

Output:



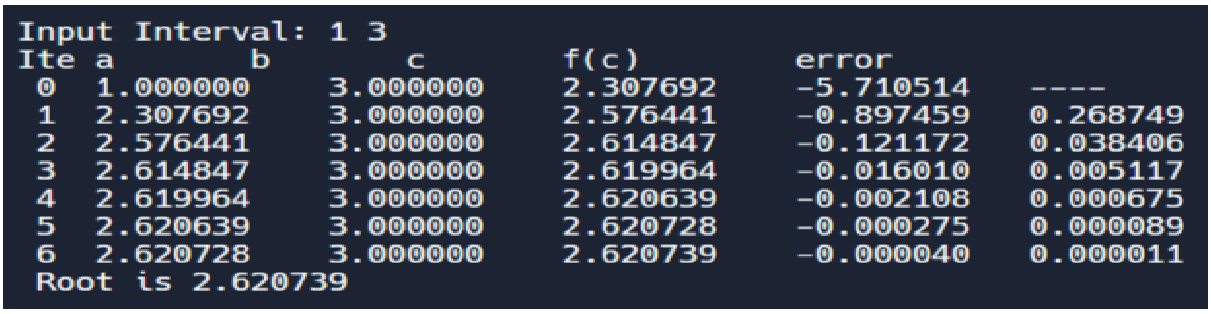
```
Input Interval: 1 3
Ite a        b        m        f(m)      error
 0  1.000000  3.000000  2.000000  -10.000000  ----
 1  2.000000  3.000000  2.500000  -2.375000  0.500000
 2  2.500000  3.000000  2.750000  2.796875   0.250000
 3  2.500000  2.750000  2.625000  0.087891   0.125000
 4  2.500000  2.625000  2.562500  -1.173584  0.062500
 5  2.562500  2.625000  2.593750  -0.550446  0.031250
 6  2.593750  2.625000  2.609375  -0.233189  0.015625
 7  2.609375  2.625000  2.617188  -0.073128  0.007812
 8  2.617188  2.625000  2.621094  0.007261   0.003906
 9  2.617188  2.621094  2.619141  -0.032963  0.001953
10  2.619141  2.621094  2.620117  -0.012859  0.000977
11  2.620117  2.621094  2.620605  -0.002802  0.000488
12  2.620605  2.621094  2.620850  0.002230   0.000244
13  2.620605  2.620850  2.620728  -0.000286  0.000122
14  2.620728  2.620850  2.620789  0.000973   0.000061
15  2.620728  2.620789  2.620758  0.000343   0.000031
Approximate Root is 2.620758
```

<u>False Position Method:</u>

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define f(x) ((x*x*x)-18)
int main(){
float a=0,b=0,error=0,m,mod;
int i=0;
printf("Input Interval: ");
scanf("%f %f",&a,&b);
if((f(a)*f(b))>0){
  printf("Invalid Interval Exit!");
  exit(1);
}
else if(f(a)==0 || f(b)==0){
  printf("Root is one of interval bounds. Root is %f\n",f(a)==0?a:b);
  exit(0);
}
printf("Ite\ta\t\tb\t\tc\t\tf(c)\t\terror\n");
do{
  mod=m;
  m=(((a*f(b))-(b*f(a)))/(f(b)-f(a)));
  printf("%2d\t%4.6f\t%4.6f\t%4.6f\t%4.6f\t",i++,a,b,m,f(m));
  if(f(m)==0){
    break;
  }else if(f(a)*f(m)<0){
    b=m;
  }else a=m;
    error=fabs(m-mod);
  if(i==1){
    printf("----\n");
  }else printf("%4.6f\n",error);
}while(error>0.00005);
printf(" Root is %4.6f \n",m);
return 0;
}
```

Output:



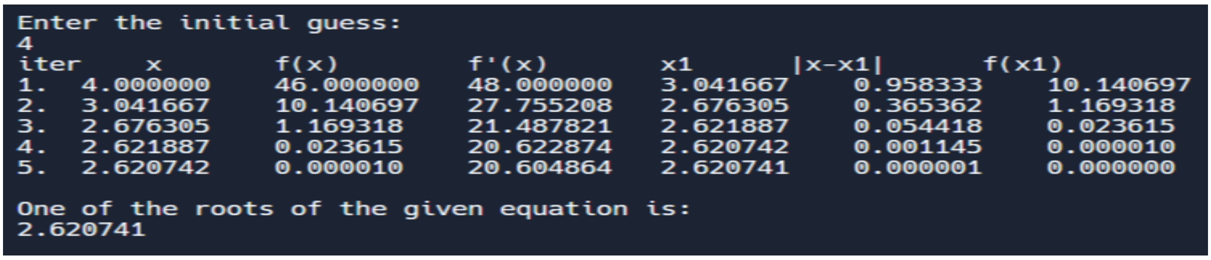
```
Input Interval: 1 3
Ite a        b        c        f(c)           error
 0  1.000000    3.000000    2.307692    -5.710514    ----
 1  2.307692    3.000000    2.576441    -0.897459    0.268749
 2  2.576441    3.000000    2.614847    -0.121172    0.038406
 3  2.614847    3.000000    2.619964    -0.016010    0.005117
 4  2.619964    3.000000    2.620639    -0.002108    0.000675
 5  2.620639    3.000000    2.620728    -0.000275    0.000089
 6  2.620728    3.000000    2.620739    -0.000040    0.000011
 Root is 2.620739
```

<u>Newton Raphson Method:</u>

Code:

```
#include<stdio.h>
#include<math.h>
double f(double x){
  return x*x*x-18;
}
double df(double x){
  return 3*x*x;
}
double rootNR(double f(double x),double df(double x),double x1,double eps,double maxSteps){
  double x;
  int i=1;
  do{
    x=x1;
    if(fabs(df(x))>=0.000000001){
      x1=x-f(x)/df(x);
      i++;
    }
  }while(fabs(x-x1)>=eps&&i<=maxSteps);
  return x1;
}
double printNR(double f(double x),double df(double x),double x1,double eps,double maxSteps){
  double x;
  int iter=1;
  printf("iter\tx\t\tf(x)\t\tf'(x)\t\tx1\t\t|x-x1|\t\tf(x1)\n");
  do{
    x=x1;
    if(fabs(df(x))>=0.000000001){
      x1=x-f(x)/df(x);
    printf("%d.\t%lf\t%lf\t%lf\t%lf\t%lf\t%lf\n",iter,x,f(x),df(x),x1,fabs(x-x1),f(x1));
    iter++;
   }
  }while(fabs(x-x1)>=eps&&iter<=maxSteps);
  return x1;
}
int main(){
  double x,x1;
  printf("Enter the initial guess:\n");
  scanf("%lf",&x);
printf("\nOne of the roots of the given equation is:\n%lf\n\n\n",printNR(f,df,x, 0.00005, 20));
return 0;
}
```

Output:



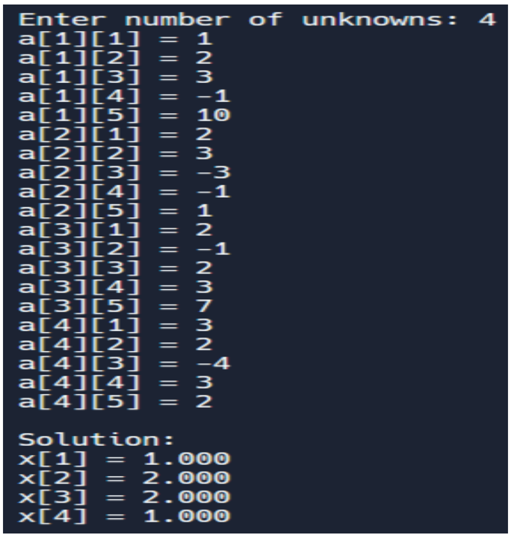
```
Enter the initial guess:
4
iter    x          f(x)        f'(x)        x1        |x-x1|       f(x1)
1.   4.000000   46.000000   48.000000   3.041667    0.958333    10.140697
2.   3.041667   10.140697   27.755208   2.676305    0.365362    1.169318
3.   2.676305   1.169318    21.487821   2.621887    0.054418    0.023615
4.   2.621887   0.023615    20.622874   2.620742    0.001145    0.000010
5.   2.620742   0.000010    20.604864   2.620741    0.000001    0.000000

One of the roots of the given equation is:
2.620741
```

<u>Gauss Elimination:</u>

Code:

```
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
#define   SIZE   20
int main()
{
  float a[SIZE][SIZE], x[SIZE], ratio;
  int i,j,k,n;
  printf("Enter number of unknowns: ");
  scanf("%d", &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]);
    }
  }
  for(i=1;i<=n-1;i++)
  {
    if(a[i][i] == 0.0)
    {
      printf("Mathematical Error!");
      exit(0);
    }
    for(j=i+1;j<=n;j++)
    {
      ratio = a[j][i]/a[i][i];
      for(k=1;k<=n+1;k++)
      {
        a[j][k] = a[j][k] - ratio*a[i][k];
      }
    }
  }
  x[n] = a[n][n+1]/a[n][n];
  for(i=n-1;i>=1;i--)
  {
    x[i] = a[i][n+1];
    for(j=i+1;j<=n;j++)
    {
       x[i] = x[i] - a[i][j]*x[j];
    }
    x[i] = x[i]/a[i][i];
  }
  printf("\nSolution:\n");
  for(i=1;i<=n;i++)
  {
    printf("x[%d] = %0.3f\n",i, x[i]);
  }
  return(0);
}
```

Output:



```
Enter number of unknowns: 4
a[1][1] = 1
a[1][2] = 2
a[1][3] = 3
a[1][4] = -1
a[1][5] = 10
a[2][1] = 2
a[2][2] = 3
a[2][3] = -3
a[2][4] = -1
a[2][5] = 1
a[3][1] = 2
a[3][2] = -1
a[3][3] = 2
a[3][4] = 3
a[3][5] = 7
a[4][1] = 3
a[4][2] = 2
a[4][3] = -4
a[4][4] = 3
a[4][5] = 2

Solution:
x[1] = 1.000
x[2] = 2.000
x[3] = 2.000
x[4] = 1.000
```

Gauss Seidel:

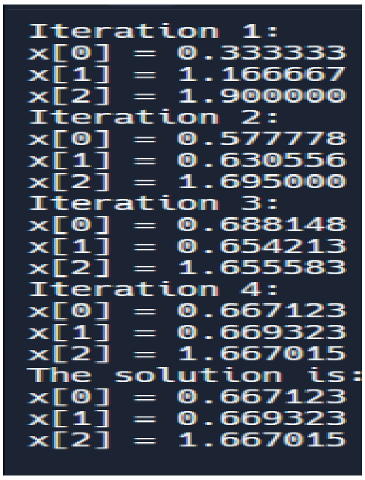
Code:

```
#include <stdio.h>
#include <math.h>
#define N 3
```

```
double* gauss_seidel(double a[N][N], double b[N], double x[N], double e, int maxit);
int main()
{
    double a[N][N] = {{3, 1, -1}, {1, 4, 1}, {2, -1, 5}};
    double b[N] = {1, 5, 9};
    double x[N] = {0, 0, 0};
    double e = 0.0001;
    int maxit = 100;
    double* ans;
    ans = gauss_seidel(a, b, x, e, maxit);
    printf("The solution is:\n");
    for (int i = 0; i < N; i++)
    {
        printf("x[%d] = %f\n", i, ans[i]);
    }
    return 0;
}
double* gauss_seidel(double a[N][N], double b[N], double x[N], double e, int maxit)
{
    double err;
    int it;
    double sum;
    double xold[N];
    do
    {
        err = 0;
        it++;
        printf("Iteration %d:\n", it);
        for (int i = 0; i < N; i++)
        {
            sum = 0;
            xold[i] = x[i];
            for (int j = 0; j < N; j++)
            {
                if (j != i)
                {
                    sum += a[i][j] * x[j];
                }
            }
            x[i] = (b[i] - sum) / a[i][i];
            printf("x[%d] = %f\n", i, x[i]);
            if (fabs(x[i] - xold[i]) > err)
            {
                err = fabs(x[i] - xold[i]);
            }
        }
    } while (err > e && it < maxit);
    return x;
```

}

Output:



```
Iteration 1:
x[0] = 0.333333
x[1] = 1.166667
x[2] = 1.900000
Iteration 2:
x[0] = 0.577778
x[1] = 0.630556
x[2] = 1.695000
Iteration 3:
x[0] = 0.688148
x[1] = 0.654213
x[2] = 1.655583
Iteration 4:
x[0] = 0.667123
x[1] = 0.669323
x[2] = 1.667015
The solution is:
x[0] = 0.667123
x[1] = 0.669323
x[2] = 1.667015
```

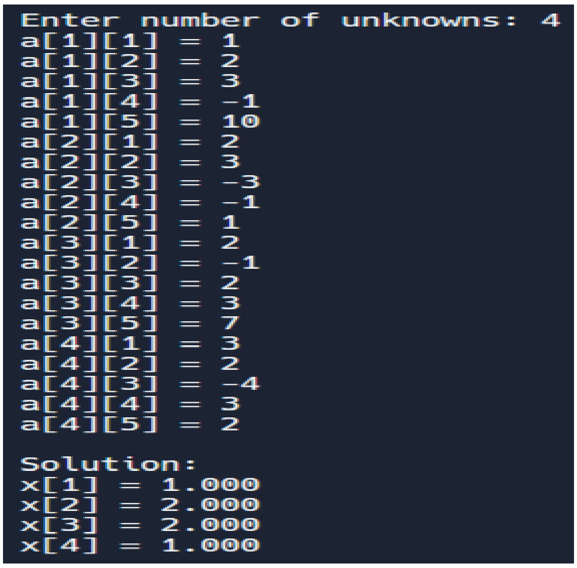
<u>Gauss Jordan:</u>

Code:

```
#include<stdio.h>
#include<math.h>
#define SIZE 20
int main()
{
    float a[SIZE][SIZE], x[SIZE], ratio;
    int i,j,k,n;
    printf("Enter number of unknowns: ");
    scanf("%d", &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]);
      }
    }
    for(i=1;i<=n;i++)
    {
      if(a[i][i] == 0.0)
      {
        printf("Mathematical Error!");
        exit(0);
      }
      for(j=1;j<=n;j++)
      {
        if(i!=j)
        {
          ratio = a[j][i]/a[i][i];
          for(k=1;k<=n+1;k++)
```

```
            {
              a[j][k] = a[j][k] - ratio*a[i][k];
            }
          }
        }
      }
      for(i=1;i<=n;i++)
      {
        x[i] = a[i][n+1]/a[i][i];
      }
      printf("\nSolution:\n");
      for(i=1;i<=n;i++)
      {
        printf("x[%d] = %0.3f\n",i, x[i]);
      }
      return(0);
}
```

Output:



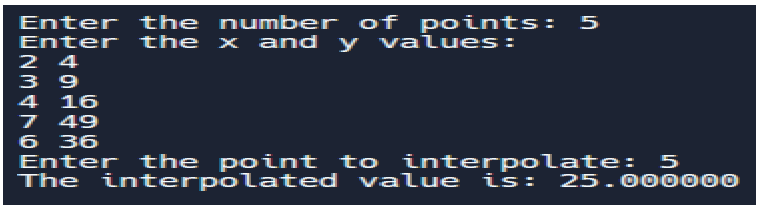
```
Enter number of unknowns: 4
a[1][1] = 1
a[1][2] = 2
a[1][3] = 3
a[1][4] = -1
a[1][5] = 10
a[2][1] = 2
a[2][2] = 3
a[2][3] = -3
a[2][4] = -1
a[2][5] = 1
a[3][1] = 2
a[3][2] = -1
a[3][3] = 2
a[3][4] = 3
a[3][5] = 7
a[4][1] = 3
a[4][2] = 2
a[4][3] = -4
a[4][4] = 3
a[4][5] = 2

Solution:
x[1] = 1.000
x[2] = 2.000
x[3] = 2.000
x[4] = 1.000
```

Lagrange Interpolation:

Code:

```
#include <stdio.h>
double lagrange(int n, double x[], double y[], double a) {
  double result = 0;
  for (int i = 0; i < n; i++) {
    double term = y[i];
    for (int j = 0; j < n; j++) {
      if (j != i) {
        term = term * (a - x[j]) / (x[i] - x[j]);
      }
    }
    result += term;
  }
  return result;
}
int main() {
  int n;
  printf("Enter the number of points: ");
  scanf("%d", &n);
  double x[n], y[n];
  printf("Enter the x and y values:\n");
  for (int i = 0; i < n; i++) {
    scanf("%lf %lf", &x[i], &y[i]);
  }
  double a;
  printf("Enter the point to interpolate: ");
  scanf("%lf", &a);
  double f = lagrange(n, x, y, a);
  printf("The interpolated value is: %lf\n", f);
  return 0;
}
```

Output:



```
Enter the number of points: 5
Enter the x and y values:
2 4
3 9
4 16
7 49
6 36
Enter the point to interpolate: 5
The interpolated value is: 25.000000
```

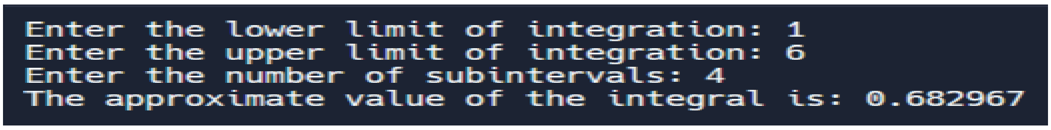
<u>Trapezoidal Rule of Integration:</u>

Code:

```
#include <stdio.h>
#include <math.h>
#define f(x) (1 / (1 + pow(x, 2)))
float trapezoidal(float a, float b, int n);
int main()
{
    float a, b;
    int n;
    float ans;
    printf("Enter the lower limit of integration: ");
    scanf("%f", &a);
    printf("Enter the upper limit of integration: ");
    scanf("%f", &b);
    printf("Enter the number of subintervals: ");
    scanf("%d", &n);
    ans = trapezoidal(a, b, n);
    printf("The approximate value of the integral is: %f\n", ans);
    return 0;
}
```

```
float trapezoidal(float a, float b, int n)
{
   float h, x, sum = 0;
   int i;
   h = (b - a) / n;
   sum += f(a) + f(b);
   for (i = 1; i < n; i++)
   {
      x = a + i * h;
      sum += 2 * f(x);
   }
   sum *= h / 2;
   return sum;
}
```

Output:



```
Enter the lower limit of integration: 1
Enter the upper limit of integration: 6
Enter the number of subintervals: 4
The approximate value of the integral is: 0.682967
```

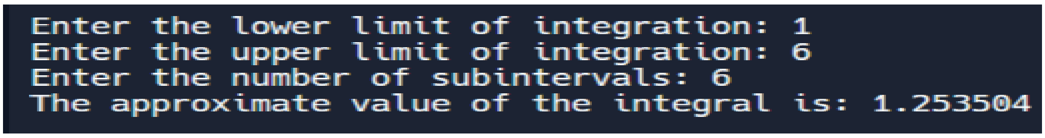
Simpson ⅓ Rule of Integration:

Code:

```
#include <stdio.h>
#include <math.h>
#define f(x) (1 / (1 + x))
float simpson_13(float a, float b, int n);
int main()
{
   float a, b;
   int n;
   float ans;
   printf("Enter the lower limit of integration: ");
   scanf("%f", &a);
   printf("Enter the upper limit of integration: ");
   scanf("%f", &b);
   printf("Enter the number of subintervals: ");
   scanf("%d", &n);
   ans = simpson_13(a, b, n);
   printf("The approximate value of the integral is: %f\n", ans);
   return 0;
}
float simpson_13(float a, float b, int n)
{
   float h, x, sum = 0;
   int i;
   if (n % 2 != 0)
   {
      printf("n must be even for Simpson's 1/3 rule\n");
```

```
        return 0;
    }
    h = (b - a) / n;
    sum += f(a) + f(b);
    for (i = 1; i < n; i += 2)
    {
        x = a + i * h;
        sum += 4 * f(x);
    }
    for (i = 2; i < n; i += 2)
    {
        x = a + i * h;
        sum += 2 * f(x);
    }
    sum *= h / 3;
    return sum;
}
```

Output:



```
Enter the lower limit of integration: 1
Enter the upper limit of integration: 6
Enter the number of subintervals: 6
The approximate value of the integral is: 1.253504
```

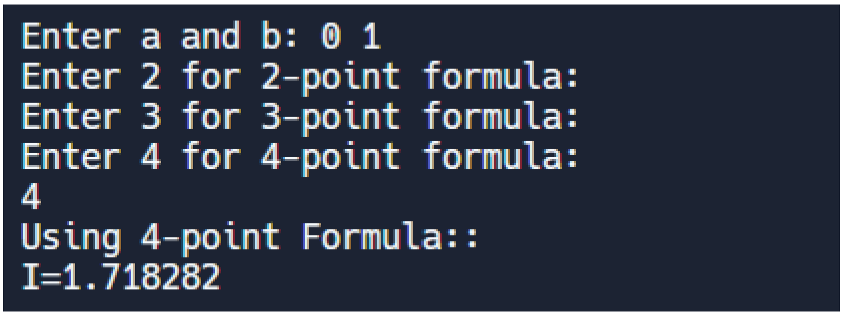
Gauss Legendre Method:

Code:

```
#include <stdio.h>
#include <math.h>
void GaussLegendre(float,float,int);
float f(float x){return (exp(x));}
float g(float a,float b,float z)
{float x=(b-a)/2*z+(b+a)/2;
return (exp(x));}
int main()
{
float a,b;
int n;
printf("Enter a and b: ");
scanf("%f%f",&a,&b);
printf("Enter 2 for 2-point formula: \n");
printf("Enter 3 for 3-point formula: \n");
printf("Enter 4 for 4-point formula: \n");
scanf("%d",&n);
switch(n)
{
  case 2:
    printf("Using 2-point Formula::\n");
    GaussLegendre(a,b,n);
    break;
  case 3:
    printf("Using 3-point Formula::\n");
    GaussLegendre(a,b,n);
    break;
  case 4:
    printf("Using 4-point Formula::\n");
    GaussLegendre(a,b,n);
    break;
  default:
    printf("INVALID\n");
    break;
}
return 0;
}
void GaussLegendre(float a,float b,int n)
{
float I;
if(a==-1 && b==1)
{
  if(n==2)
  {
  I=1*f(-1/sqrt(3))+1*f(1/sqrt(3));
  printf("I=%f",I);
  }
  if(n==3)
  {
  I=5/9*f(-sqrt(3/5))+8/9*f(0)+5/9*f(sqrt(3/5));
  printf("I=%f",I);
  }
  if(n==4)
  {
I=0.34785*f(-0.86114)+0.65215*f(-0.33998)+0.65215*f(0.33998)+0.34785*f(0.86114);
  printf("I=%f",I);
  }
}
else
{
  if(n==2)
  {
  I=(b-a)/2*(1*g(a,b,-1/sqrt(3))+1*g(a,b,1/sqrt(3)));
  printf("I=%f",I);
  }
  if(n==3)
  {
I=(b-a)/2*(5/9*g(a,b,-sqrt(3/5))+8/9*g(a,b,0)+5/9*g(a,b,sqrt(3/5)));
  printf("I=%f",I);
  }
  if(n==4)
```

```
 {
I=(b-a)/2*(0.34785*g(a,b,-0.86114)+0.65215*g(a,b,-0.33998)+0.65215*g(a,b,0.33998)+0.34785*g(a,b,0.86114));
 printf("I=%f",I);
 }
}
}
```

Output:



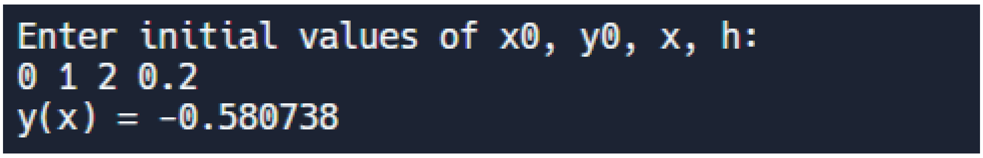
```
Enter a and b: 0 1
Enter 2 for 2-point formula:
Enter 3 for 3-point formula:
Enter 4 for 4-point formula:
4
Using 4-point Formula::
I=1.718282
```

Heun's method:

Code:

```
#include <stdio.h>
float dydx(float x, float y) { return (x*x + y - 2); }
float heuns(float x0, float y0, float x, float h)
{
 int n = (int)((x - x0) / h);
 float k1, k2;
 float y = y0;
 for (int i = 1; i <= n; i++) {
  k1 = h * dydx(x0, y);
  k2 = h * dydx(x0 + h, y + k1);
  y = y + (k1 + k2)/2;
  x0 = x0 + h;
 }
 return y;
}
int main()
{
 float x0, y0, x, h;
 printf("Enter initial values of x0, y0, x, h:\n");
 scanf("%f %f %f %f", &x0, &y0, &x, &h);
 printf("y(x) = %f", heuns(x0, y0, x, h));
 return 0;
}
```

Output:



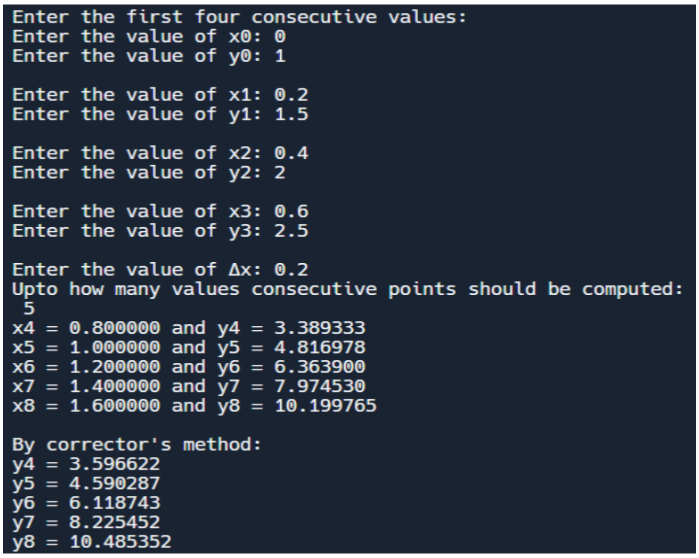
```
Enter initial values of x0, y0, x, h:
0 1 2 0.2
y(x) = -0.580738
```

Milne’s Predictor and corrector Method:

Code:

```
#include <stdio.h>
#include <stdlib.h>
#define MAX        15
float dybydx(float x, float y);
void milnep(float x[], float y[], float delx, int n, float ydash[]);
void milnec(float x[], float y[], float delx, float ydash[], int n);
int main()
{
  int         i = 0, n;
  float       x[MAX], y[MAX], ydash[MAX];
  float       delx;
  printf("Enter the first four consecutive values: \n");
  while (i < 4)
  {
    printf("Enter the value of x%d: ", i);
    scanf("%f", &x[i]);
    printf("Enter the value of y%d: ", i);
    scanf("%f", &y[i]);
    printf("\n");
    i++;
  }
  printf("Enter the value of Δx: ");
  scanf("%f", &delx);
  printf("Upto how many values consecutive points should be computed: ");
  scanf("%d", &n);
  milnep(x, y, delx, n+4, ydash);
  milnec(x, y, delx, ydash, n+4);
  exit(0);
}
void milnep(float x[], float y[], float delx, int n, float ydash[])
{
  int       i = 4, j;
  for (j = 0; j < 4; j++)
  {
    ydash[j] = dybydx(x[j], y[j]);
  }
  while (i < n)
  {
    y[i] = y[i-4] + (4*delx*(2*ydash[i-3]-ydash[i-2]+2*ydash[i-1]))/3;
    x[i] = x[i-1] + delx;
    ydash[i] = dybydx(x[i], y[i]);
    printf("x%d = %f and y%d = %f\n", i, x[i], i, y[i]);
    i++;
  }
  return ;
}
void milnec(float x[], float y[], float delx, float ydash[], int n)
{
  int       i = 4;
  float     ycheck;
  printf("\nBy corrector's method: \n");
  while (i < n)
  {
    ycheck = y[i-2] + delx*(ydash[i-2]+4*ydash[i-1]+ydash[i])/3;
    printf("y%d = %f\n", i, ycheck);
    i++;
  }
  return ;
}
float dybydx(float x, float y)
{
  float      fx;
  fx = y+3*x-x*x;
  return fx;
}
```

Output:



```
Enter the first four consecutive values:
Enter the value of x0: 0
Enter the value of y0: 1

Enter the value of x1: 0.2
Enter the value of y1: 1.5

Enter the value of x2: 0.4
Enter the value of y2: 2

Enter the value of x3: 0.6
Enter the value of y3: 2.5

Enter the value of Δx: 0.2
Upto how many values consecutive points should be computed:
 5
x4 = 0.800000 and y4 = 3.389333
x5 = 1.000000 and y5 = 4.816978
x6 = 1.200000 and y6 = 6.363900
x7 = 1.400000 and y7 = 7.974530
x8 = 1.600000 and y8 = 10.199765

By corrector's method:
y4 = 3.596622
y5 = 4.590287
y6 = 6.118743
y7 = 8.225452
y8 = 10.485352
```