GAUSS ELIMINATION:

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#include<stdlib.h>
#define    SIZE    10
int main()
{  float a[SIZE][SIZE], x[SIZE], ratio; int i,j,k,n;
      /* 1. Reading number of unknowns */
      printf("Enter number of unknowns: ");
      scanf("%d", &n);
      /* 2. Reading Augmented Matrix */
      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);}
```

GAUSS JORDAN:

```
#include<stdio.h>
int main()
{
    int i,j,k,n;
    float A[20][20],c,x[10];
    printf("\nEnter the size of matrix: ");
    scanf("%d",&n);
    printf("\nEnter the elements of augmented matrix
row-wise:\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(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];}}}

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

NEWTON FORWARD INTERPOLATION:

```
#include<stdio.h>
  int main() {
   int n;
   printf("Enter the number of terms\n");
   scanf("%d",&n);
   float diff[n][n+1];
    printf("enter the values of x");
   for(int i=0;i<n;i++){
      scanf("%f",&diff[i][0]);
   }
    printf("enter the values of y");
   for(int i=0;i<n;i++){
      scanf("%f",&diff[i][1]);
   }
   for(int j=2; j<=n;j++){
      for(int i=n-1; i>=j-1;i--){
         diff[i][j] = diff[i][j-1]-diff[i-1][j-1];
      }
   }
   for(int i=0;i<n;i++){
      for(int j=0;j<=i+1;j++){
         printf("%0.6f \t",diff[i][j]);
      }
      printf("\n");
   }
   float x;
   printf("enter the value of x");
   scanf("%f",&x);
   float h= diff[1][0]- diff[0][0] ;
   float fact =1,p= (x-diff[n-1][0])/h ,u=p;
   float res= diff[n-1][1];
   for(int i=2;i<=n;i++){
      res+= ((p)*diff[n-1][i])/fact;
      fact*= i;
      p= (p)*(u+(i-1));
   }
   printf("result is %f", res);
  return 0;
}
```

NEWTON BACKWARD INTERPOLATION:

```
#include <stdio.h>
int main()
{
   int n;
   printf("Enter the number of terms\n");
   scanf("%d",&n);
   float diff[n][n+1];
   for(int i=0;i<n;i++){
      scanf("%f",&diff[i][0]);
   }
   for(int i=0;i<n;i++){
      scanf("%f",&diff[i][1]);
   }
   for(int j=2; j<=n;j++){
      for(int i=0; i<n-j+1;i++){
         diff[i][j] = diff[i+1][j-1]-diff[i][j-1];
      }
   }
   for(int i=0;i<n;i++){
      for(int j=0;j<n-i+1;j++){
         printf("%0.6f \t",diff[i][j]);
      }
      printf("\n");
   }
   float x;
   printf("enter the value of x");
   scanf("%f",&x);
   float h= diff[1][0]- diff[0][0] ;
   float fact =1,p= (x-diff[0][0])/h ,u=p;
   float res= diff[0][1];
   for(int i=2;i<=n;i++){
      res+= ((p)*diff[0][i])/fact;
      fact*= i;
      p= (p)*(u-(i-1));
   }
   printf("result is %f", res);
   return 0;
}
```

LAGRANGE INTERPOLATION METHOD

```
#include<stdio.h>
#include<conio.h>
void main()
{
	float x[100], y[100], xp, yp=0, p;
	int i,j,n;
	printf("Enter number of data: ");
	scanf("%d", &n);
	printf("Enter data:\n");
	for(i=1;i<=n;i++){
		printf("x[%d] = ", i);
		scanf("%f", &x[i]);
		printf("y[%d] = ", i);
		scanf("%f", &y[i]);}
	printf("Enter interpolation point: ");
	scanf("%f", &xp);
	for(i=1;i<=n;i++)
	{
		p=1;
		for(j=1;j<=n;j++)
		{
			if(i!=j)
			{
				p = p* (xp - x[j])/(x[i] - x[j]);
			}
		}
		yp = yp + p * y[i];
	}
	printf("Interpolated value at %.3f is %.3f.", xp, yp);
}
```

TRAPEZOIDAL

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
/* Define function here */
#define f(x) 1/(1+pow(x,2))
int main()
{
 float lower, upper, integration=0.0, stepSize, k;
 int i, subInterval;
 printf("Enter lower limit of integration: ");
 scanf("%f", &lower);
 printf("Enter upper limit of integration: ");
 scanf("%f", &upper);
 printf("Enter number of subintervals: ");
 scanf("%d", &subInterval);

 stepSize = (upper - lower)/subInterval;

 /* Finding Integration Value */
 integration = f(lower) + f(upper);
 for(i=1; i<= subInterval-1; i++){
  k = lower + i*stepSize;
  integration = integration + 2 * f(k);}
 integration = integration * stepSize/2;
 printf("\nRequired value of integration is: %.3f", integration);
 return 0;
}
```

SIMPSON ⅓ RULE

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
/* Define function here */
#define f(x) 1/(1+x*x)
int main()
{
 float lower, upper, integration=0.0, stepSize, k;
 int i, subInterval;
 printf("Enter lower limit of integration: ");
 scanf("%f", &lower);
 printf("Enter upper limit of integration: ");
 scanf("%f", &upper);
 printf("Enter number of sub intervals: ");
 scanf("%d", &subInterval);
 stepSize = (upper - lower)/subInterval;

 /* Finding Integration Value */
 integration = f(lower) + f(upper);
 for(i=1; i<= subInterval-1; i++)
 {
  k = lower + i*stepSize;
  if(i%2==0)
   integration = integration + 2 * f(k);
  else
   integration = integration + 4 * f(k);
 }
 integration = integration * stepSize/3;
 printf("\nRequired value of integration is: %.3f", integration);
 return 0;}
```

SIMPSON ⅜ RULE

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#define f(x) 1/(1+x*x)
int main()
{
 float lower, upper, integration=0.0, stepSize, k;
 int i, subInterval;
 printf("Enter lower limit of integration: ");
 scanf("%f", &lower);
 printf("Enter upper limit of integration: ");
 scanf("%f", &upper);
 printf("Enter number of sub intervals: ");
 scanf("%d", &subInterval);
 stepSize = (upper - lower)/subInterval;
 integration = f(lower) + f(upper);
 for(i=1; i<= subInterval-1; i++)
 {
  k = lower + i*stepSize;
  if(i%3 == 0)
   integration = integration + 2 * f(k);
```

```
 else
  integration = integration + 3 * f(k);
}
integration = integration * stepSize*3/8;
printf("\nRequired value of integration is: %.3f", integration);
Return 0;
```

EULER METHOD:

```
#include<stdio.h>
#include<conio.h>
#define f(x,y) x+y
int main()
{
 float x0, y0, xn, h, yn, slope;
 int i, n;
 printf("Enter Initial Condition\n");
 printf("x0 = ");
 scanf("%f", &x0);
 printf("y0 = ");
 scanf("%f", &y0);
 printf("Enter calculation point xn = ");
 scanf("%f", &xn);
 printf("Enter number of steps: ");
```

```
 scanf("%d", &n);
 h = (xn-x0)/n;
 printf("\nx0\ty0\tslope\tyn\n");
 printf("----------------------------\n");
 for(i=0; i < n; i++)
 {
  slope = f(x0, y0);
  yn = y0 + h * slope;
  printf("%.4f\t%.4f\t%0.4f\t%.4f\n",x0,y0,slope,yn);
  y0 = yn;
  x0 = x0+h;
 }
 /* Displaying result */
 printf("\nValue of y at x = %0.2f is %0.3f",xn, yn);
 return 0;
}
```

RK METHOD

```
#include<stdio.h>
#include<conio.h>
#define f(x,y) (y*y-x*x)/(y*y+x*x)
int main()
{
 float x0, y0, xn, h, yn, k1, k2, k3, k4, k;
 int i, n;
 printf("Enter Initial Condition\n");
 printf("x0 = ");
 scanf("%f", &x0);
 printf("y0 = ");
 scanf("%f", &y0);
 printf("Enter calculation point xn = ");
 scanf("%f", &xn);
 printf("Enter number of steps: ");
 scanf("%d", &n);
 h = (xn-x0)/n;
 printf("\nx0\ty0\tyn\n");
 for(i=0; 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)));
  k = (k1+2*k2+2*k3+k4)/6;
  yn = y0 + k;
  printf("%0.4f\t%0.4f\t%0.4f\n",x0,y0,yn);
  x0 = x0+h;
  y0 = yn;
 }

 /* Displaying result */
 printf("\nValue of y at x = %0.2f is %0.3f",xn, yn);
 return 0;
}
```