

```

1 // 01 01 Bisection Method
2 #include <stdio.h> //input 1, 3
3 #include <stdlib.h>
4 #include <math.h>
5 #define f(x) ((x*x*x)-18)
6 int main()
7 {
8     float a=0,b=0,error=0,m,mold;
9     int i=0;
10    printf("Input Interval: ");
11    scanf("%f %f",&a,&b);
12    if((f(a)*f(b))>0)
13    {
14        printf("Invalid Interval
Exit!"); //to test whether search
interval is okay or not
15        exit(1);
16    }
17    else if(f(a)==0 || f(b)==0)
18    {
19        printf("Root is one of interval
bounds. Root is %f\n",f(a)==0?a:b);
20        exit(0);
21    }
22
23    printf("Ite\ta\t\tb\t\tm\t\tf(m)\t\terror\n"
);
24    do{
25        mold=m;
26        m=(a+b)/2;
27
28        printf("%2d\t%4.6f\t%4.6f\t%4.6f\t%4.6f\t",i
++,a,b,m,f(m));
29        if(f(m)==0)
30        {
31            printf("Root is %4.6f\n",m);
32        }
33        else if((f(a)*f(m))<0)
34        {
35            b=m;
36        }
37        else a=m;

```

```
36         error=fabs (m-mold) ;
37     if (i==1)
38     {
39         printf ("----\n") ;
40     }
41     else
42         printf ("%4.6f\n",error) ;
43 } while (error>0.00005) ;
44 printf ("Approximate Root is %4.6f",m) ;
45 return 0 ;
46 }
47
```

```

1 // 01 02 Regula Falsi
2 #include <stdio.h> //input 1, 3
3 #include <stdlib.h>
4 #include <math.h>
5 #define f(x) ((x*x*x)-18)
6 int main()
7 {
8     float a=0,b=0,error=0,c,cold;
9     int i=0;
10    printf("Input Interval: ");
11    scanf("%f %f",&a,&b);
12    if((f(a)*f(b))>0)
13    {
14        printf("Invalid Interval Exit!");
15        exit(1);
16    }
17    else if(f(a)==0 || f(b)==0)
18    {
19        printf("Root is one of interval
20    bounds. Root is %f\n",f(a)==0?a:b);
21        exit(0);
22    }
23
24    printf("Ite\ta\t\tb\t\tc\t\tf(c)\t\terror\n"
25    );
26
27    do{
28        cold=c;
29
30        c=((a*f(b))-(b*f(a)))/(f(b)-f(a));
31
32        printf("%2d\t%4.6f\t%4.6f\t%4.6f\t%4.6f\t",i
33        ++,a,b,c,f(c));
34
35        if(f(c)==0)
36        {
37            break;
38        }
39        else if(f(a)*f(c)<0)
40        {
41            b=c;
42        }
43        else a=c;
44        error=fabs(c-cold);

```

```
37         if (i==1)
38         {
39             printf ("----\n");
40         }
41         else
42             printf ("%4.6f\n", error);
43
44     } while (error>0.00005);
45     printf (" Root is %4.6f \n", c);
46     return 0;
47 }
48
```

```

1 // 01 03 Newton Raphson
2 #include <stdio.h> //input 2
3 #include <stdlib.h>
4 #include <math.h>
5 #define f(x) ((x*x*x)-18)
6 #define fd(x) (3*x*x)
7 #define fdd(x) 6*x
8 int main()
9 {
10     float
x0,x1,error,errorold,converge,order;
11     int i=0;
12     printf("Input the approximation : ");
13     scanf("%f",&x0);
14
converge=(f(x0)*fdd(x0))/(fd(x0)*fd(x0));
15     if(converge >1)
16         exit(1);
17
printf("Ite\tX0\t\tX1\t\tError\t\tOrder\n");
18     do{
19         errorold=error;
20         x1=x0-(f(x0)/fd(x0));
21         if(f(x1)==0)
22         {
23             break;
24         }
25         error=fabs(x1-x0);
26
printf("%2d\t%4.6f\t%4.6f\t%4.6f\t",++i,x0,x
1,error);
27         if(i==1||error==0||errorold==1)
28         {
29             printf("----\n");
30         }
31         else
32         {
33             order=log(error)/log(errorold);
34             printf("%4.6f\n",order);
35         }
36         x0=x1;
37     }while(error>0.00005);

```

```
38         printf("Root is %4.6f", x0);  
39         return 0;  
40     }  
41
```

```

1 // 01_04_Secant_Method
2 #include<stdio.h>
3 #include<math.h>
4 /*Function whose root is to be determined*/
5 double f(double x)
6 {
7     return x*x-4;
8 }
9 main()
10 {
11     int iter=1, maxSteps;
12     double x1,x2,x3,eps;
13     printf("Enter the accuracy desired:
14 \n");
15     scanf("%lf",&eps);
16     printf("Enter the initial guesses: \nx1
17 = ");
18     scanf("%lf",&x1);
19     printf("x2 = ");
20     scanf("%lf",&x2);
21     printf("Enter the max number of
22 iterations to be performed: ");
23     scanf("%d",&maxSteps);
24
25     printf("\titer\t\ttx1\t\ttx2\t\ttx3\t\ttf(x3)\n");
26
27     printf("_____
28 _____\n");
29
30     do
31     {
32         x3=(
33         x1*f(x2)-x2*f(x1))/(f(x2)-f(x1));
34
35         printf("%d\t%lf\t%lf\t%lf\t%lf\n",iter,x1,x2
36 ,x3,f(x3));
37
38         x1=x2;
39         x2=x3;
40         iter++;
41     }
42     while( fabs(f(x3))>eps&&iter<=maxSteps);
43     printf("\nOne of the roots is: %lf",x3);
44 }
```

```

1 // 01 05 Gauss Elimination
2 #include<stdio.h>
3 #include<conio.h>
4 #include<math.h>
5 #define SIZE 10
6 int main()
7 {
8     float a[SIZE][SIZE], x[SIZE], ratio;
9     int i,j,k,n;
10    /* Inputs */
11    /* 1. Reading number of unknowns */
12    printf("Enter number of unknowns: ");
13    scanf("%d", &n);
14    /* 2. Reading Augmented Matrix */
15    for(i=1;i<=n;i++)
16    {
17        for(j=1;j<=n+1;j++)
18        {
19            printf("a[%d][%d] = ",i,j);
20            scanf("%f", &a[i][j]);
21        }
22    }
23    /* Applying Gauss Elimination */
24    for(i=1;i<=n-1;i++)
25    {
26        if(a[i][i] == 0.0)
27        {
28            printf("Mathematical Error!");
29            exit(0);
30        }
31        for(j=i+1;j<=n;j++)
32        {
33            ratio = a[j][i]/a[i][i];
34            for(k=1;k<=n+1;k++)
35            {
36                a[j][k] = a[j][k] -
ratio*a[i][k];
37            }
38        }
39    }
40    /* Obtaining Solution by Back
Subsitution */

```



```

41     x[n] = a[n][n+1]/a[n][n];
42     for(i=n-1;i>=1;i--)
43     {
44         x[i] = a[i][n+1];
45         for(j=i+1;j<=n;j++)
46         {
47             x[i] = x[i] - a[i][j]*x[j];
48         }
49         x[i] = x[i]/a[i][i];
50     }
51     /* Displaying Solution */
52     printf("\nSolution:\n");
53     for(i=1;i<=n;i++)
54     {
55         printf("x[%d] = %0.3f\n",i, x[i]);
56     }
57     getch();
58 }
59

```

```

1 // 01 06 Gauss Seidal
2 #include<stdio.h>
3 #include<conio.h>
4 #include<math.h>
5 #define f1(x,y,z) (17-y+2*z)/20
6 #define f2(x,y,z) (-18-3*x+z)/20
7 #define f3(x,y,z) (25-2*x+3*y)/20
8
9 // Arrange systems of linear
10 // equations to be solved in
11 // diagonally dominant form
12 // and form equation for each
13 // unknown and define here
14
15 // In this example we are solving
16 // 3x + 20y - z = -18
17 // 2x - 3y + 20z = 25
18 // 20x + y - 2z = 17
19
20 // Arranging given system of linear
21 // equations in diagonally dominant
22 // form:
23 // 20x + y - 2z = 17
24 // 3x + 20y - z = -18
25 // 2x - 3y + 20z = 25
26
27 // Equations:
28 // x = (17-y+2z)/20
29 // y = (-18-3x+z)/20
30 // z = (25-2x+3y)/20
31
32 // Defining function
33
34 // Main function
35 int main()
36 {
37     float x0=0, y0=0, z0=0, x1, y1, z1,
e1, e2, e3, e;
38     int count=1;
39     printf("Enter tolerable error:\n");
40     scanf("%f", &e);
41     printf("\nCount\ttx\tty\ttz\n");

```

```

42         do
43         {
44             // Calculation
45             x1 = f1(x0, y0, z0);
46             y1 = f2(x1, y0, z0);
47             z1 = f3(x1, y1, z0);
48
49             printf("%d\t%0.4f\t%0.4f\t%0.4f\n", count,
50                 x1, y1, z1);
51
52             // Error
53             e1 = fabs(x0-x1);
54             e2 = fabs(y0-y1);
55             e3 = fabs(z0-z1);
56
57             count++;
58
59             // Set value for next iteration
60             x0 = x1;
61             y0 = y1;
62             z0 = z1;
63             }while(e1>e && e2>e && e3>e);
64             printf("\nSolution: x=%0.3f, y=%0.3f
65             and z = %0.3f\n", x1, y1, z1);
66             getch();
67             return 0;
68         }

```

```

1 // 01 07 Newton Backward
2 #include<stdio.h>
3 #include<conio.h>
4
5 int main()
6 {
7     float x[20], y[20][20];
8     int i,j, n;
9     /* Input Section */
10    printf("Enter number of data?\n");
11    scanf("%d", &n);
12    printf("Enter data:\n");
13    for(i = 0; i < n ; i++)
14    {
15        printf("x[%d]=", i);
16        scanf("%f", &x[i]);
17        printf("y[%d]=", i);
18        scanf("%f", &y[i][0]);
19    }
20    /* Generating Backward Difference
21    Table */
22    for(i = 1; i < n; i++)
23    {
24        for(j = n-1; j > i-1; j--)
25        {
26            y[j][i] = y[j][i-1] -
27            y[j-1][i-1];
28        }
29    }
30    /* Displaying Backward Difference
31    Table */
32    printf("\nBACKWARD DIFFERENCE
33    TABLE\n\n");
34    for(i = 0; i < n; i++)
35    {
36        printf("%0.2f", x[i]);
37        for(j = 0; j <= i ; j++)
38        {
39            printf("\t%0.2f", y[i][j]);
40        }
41        printf("\n");
42    }

```

```
39         getch(); /* Holding Screen */
40         return 0;
41     }
42
```

```

1 // 01 08 Newton Forward
2 #include<stdio.h>
3 #include<conio.h>
4 int main()
5 {
6     float x[20], y[20][20];
7     int i, j, n;
8     // Input Section
9     printf("Enter number of data: ");
10    scanf("%d", &n);
11    printf("Enter data:\n");
12    for(i = 0; i < n ; i++)
13    {
14        printf("x[%d]=", i);
15        scanf("%f", &x[i]);
16        printf("y[%d]=", i);
17        scanf("%f", &y[i][0]);
18    }
19    // Generating Forward Difference Table
20    for(i = 1; i < n; i++)
21    {
22        for(j = 0; j < n-i; j++)
23        {
24            y[j][i] = y[j+1][i-1] -
25            y[j][i-1];
26        }
27    }
28    // Displaying Forward Difference Table
29    printf("\nFORWARD DIFFERENCE
30    TABLE\n\n");
31    for(i = 0; i < n; i++)
32    {
33        printf("%0.2f", x[i]);
34        for(j = 0; j < n-i ; j++)
35        {
36            printf("\t%0.2f", y[i][j]);
37        }
38        printf("\n");
39    }
40    getch();
41    return 0;

```

```

1 // 01 09 Newton divide difference formula
2 #include<stdio.h>
3 #include<conio.h>
4
5 void main()
6 {
7     int x[10], y[10], p[10];
8     int k,f,n,i,j=1,f1=1,f2=0;
9     printf("Enter the number of
observations: ");
10     scanf("%d", &n);
11
12     printf("Enter the different values of
x: \n");
13     for (i=1;i<=n;i++)
14         scanf("%d", &x[i]);
15
16     printf("\nThe corresponding values of
y are: \n");
17     for (i=1;i<=n;i++)
18         scanf("%d", &y[i]);
19
20     f=y[1];
21     printf("\nEnter the value of 'k' in
f(k) you want to evaluate: ");
22     scanf("%d", &k);
23
24     do
25     {
26         for (i=1;i<=n-1;i++)
27         {
28             p[i] =
((y[i+1]-y[i])/(x[i+j]-x[i]));
29             y[i]=p[i];
30         }
31         f1=1;
32         for(i=1;i<=j;i++)
33         {
34             f1*=(k-x[i]);
35         }
36         f2+=(y[1]*f1);
37         n--;

```

```
38         j++;
39     }
40
41     while (n!=1);
42     f+=f2;
43     printf("\nf(%d) = %d", k , f);
44     getch();
45 }
46
```



```

1  // 01 10 Labrange's Interpolation
2  #include<stdio.h>
3  #include<conio.h>
4  void main()
5  {
6      float x[100], y[100], xp, yp=0, p;
7      int i,j,n;
8      /* Input Section */
9      printf("Enter number of data: ");
10     scanf("%d", &n);
11     printf("Enter data:\n");
12     for(i=1;i<=n;i++)
13     {
14         printf("x[%d] = ", i);
15         scanf("%f", &x[i]);
16         printf("y[%d] = ", i);
17         scanf("%f", &y[i]);
18     }
19     printf("Enter interpolation point: ");
20     scanf("%f", &xp);
21     /* Implementing Lagrange Interpolation
22     */
23     for(i=1;i<=n;i++)
24     {
25         p=1;
26         for(j=1;j<=n;j++)
27         {
28             if(i!=j)
29             {
30                 p = p* (xp - x[j])/(x[i] -
31                 x[j]);
32             }
33             yp = yp + p * y[i];
34         }
35         printf("Interpolated value at %.3f is
36         %.3f.", xp, yp);
37         getch();
38     }
39 }

```