Q. Write a program to find the Square Roots of Quadratic Equation.

```
#include <math.h>
#include <stdio.h>

int main()
{
   double a, b, c, d, r1, r2, real, imag;
   printf("Enter coefficients a, b and c: ");
   scanf("%lf %lf %lf", &a, &b, &c);

   d = b * b - 4 * a * c;
   if (d > 0)
   {
      r1 = (-b + sqrt(d)) / (2 * a);
      r2 = (-b - sqrt(d)) / (2 * a);
      printf("r1 = %.2lf and r2 = %.2lf", r1, r2);
   }

   else if (d == 0)
   {
      r1 = r2 = -b / (2 * a);
      printf("r1 = r2 = %.2lf;", r1);
   }

   else
   {
      real = -b / (2 * a);
      imag = sqrt(-d) / (2 * a);
      printf("r1 = %.2lf+%.2lfi and r2 = %.2f-%.2fi", real, imag, real, imag);
   }

   return 0;
}
```

OUTPUT

Output Clear

```
/tmp/MyfwqoCTGz.o
Enter coefficients a, b and c: 1 5 6
r1 = -2.00 and r2 = -3.00
```

Q. Write a program in C to implement bisection method.

```
#include<stdio.h>
#include<math.h>

double F( double x)
{
   return (10 - pow(x,2));
}

int main()
{
   printf(" f(x) = 10 - x^2");
   double x0,x1;
   printf("\nEnter the first approximation to the root : ");
   scanf("%lf",&x0);

   printf("Enter the second approximation to the root : ");
   scanf("%lf",&x1);

   int iter;
   printf("Enter the number of iteration you want to perform : ");
   scanf("%d",&iter);

   int ctr = 1;
   double l1 = x0;
   double l2 = x1;
   double r,f1,f2,f3;

   if(F(l1)==0)
      r = l1;
   else if(F(l2)==0)
      r = l2;
   else
   {
      while(ctr <= iter)
      {
         f1 = F(l1);
         r = (l1+l2)/2.0;
         f2 = F(r);
         f3 = F(l2);

         if(f2 == 0)
```

```
        {
            r = f2;
            break;
        }
        printf("The root after %d iteration is %lf\n",ctr,r);

        if(f1*f2 < 0)
            l2 = r;
        else if(f2*f3 < 0)
            l1 = r;
        ctr++;
    }
}

printf("The approximation to the root is %lf\n",r);
return 0;
}
```

OUTPUT

Output Clear

```
/tmp/PcmlIOuAVW.o
f(x) = 10 - x^2
Enter the first approximation to the root : 2
Enter the second approximation to the root : -5
Enter the number of iteration you want to perform : 6
The root after 1 iteration is -1.500000
The root after 2 iteration is -3.250000
The root after 3 iteration is -2.375000
The root after 4 iteration is -2.812500
The root after 5 iteration is -3.031250
The root after 6 iteration is -3.140625
The approximation to the root is -3.140625
```

Q. Write a program in C to implement regula falsi method.

```
#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,c,flag;
      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
      {
         flag=c;
         c=(((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,c,f(c));
         if(f(c)==0)
         {
             break;
         }
        else if(f(a)*f(c)<0)
        {
             b=c;
        }
        else a=c;
        error=fabs(c-flag);
```

```
        if(i==1)
        {
            printf("----\n");
        }
        else
        printf("%4.6f\n",error);
        }
        while(error>0.00005);
        printf(" Root is %4.6f \n",c);
        return 0;
}
```

OUTPUT

Output Clear

```
/tmp/IjR7OA2aJC.o
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
```

Q. Write a program in C to implement secant method.

```
#include<stdio.h>
#include<math.h>

#define f(x) (pow(x,3)-18)

int main()
{
   float x0,x1,x2,error;
   int i=0;
   printf("Input Two Approximations: ");
   scanf("%f %f",&x0,&x1);
   printf("Ite\tX0\t\tX1\t\tf(X0)\t\tf(X1)\t\tError\n");
   do
   {
      x2=((x0*f(x1))-((x1)*f(x0)))/((f(x1)-f(x0)));
      printf("%2d\t%4.6f\t%4.6f\t%4.6f\t%4.6f\t%4.6f\n",i++,x0,x1,f(x0),f(x1),error);
      error=fabs((x2)-(x1));
      x0=x1;
      x1=x2;
   }
   while(error>0.00005);
   return 0;
}
```

OUTPUT

Output Clear

```
/tmp/IjR70A2aJC.o
Input Two Approximations: 1 2
Ite X0       X1       f(X0)        f(X1)        Error
 0  1.000000    2.000000    -17.000000  -10.000000  0.000000
 1  2.000000    3.428571    -10.000000  22.303208   1.428571
 2  3.428571    2.442238    22.303208   -3.433201   0.986333
 3  2.442238    2.573814    -3.433201   -0.949728   0.131575
 4  2.573814    2.624131    -0.949728   0.069927    0.050317
 5  2.624131    2.620680    0.069927    -0.001263   0.003451
 6  2.620680    2.620741    -0.001263   -0.000001   0.000061
```

Q. Write a program in C to implement Newton Raphson method.

```
#include<stdio.h>
#include<math.h>
float f(float x)
{
   return x*log10(x) - 1.2;
}
float df (float x)
{
   return log10(x) + 0.43429;
}
void main()
{
   int itr, maxmitr;
   float h, x0, x1, allerr;
   printf("\nEnter x0, allowed error and maximum iterations\n");
   scanf("%f %f %d", &x0, &allerr, &maxmitr);
   for (itr=1; itr<=maxmitr; itr++)
   {
      h=f(x0)/df(x0);
      x1=x0-h;
      printf("At Iteration no. %3d, x = %9.6f\n", itr, x1);
      if (fabs(h) < allerr)
      {
         printf("After %3d iterations, root = %8.6f\n", itr, x1);
         return 0;
      }
      x0=x1;
   }
   printf(" The required solution does not converge or iterations are insufficient\n");
   return 1;
}
```

OUTPUT

Output Clear

```
/tmp/kUBNSlHdT4.o
Enter x0, allowed error and maximum iterations
2 0.0001 10
At Iteration no.   1, x =  2.813170
At Iteration no.   2, x =  2.741109
At Iteration no.   3, x =  2.740646
At Iteration no.   4, x =  2.740646
After   4 iterations, root = 2.740646
```

Q. Write a program to implement Gauss Elimination Method.

```
#include<stdio.h>
int main()
{
int i,j,k,n;
float A[20][20],c,x[10],sum=0.0;

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]); }
	}
	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];
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=%f\t",i,x[i]);
}
return(0);
}
```

OUTPUT

```
/tmp/L57tXf6yYB.o
Enter the order of matrix: 3
Enter the elements of augmented matrix row-wise:

A[1][1] : 10
A[1][2] : -7
A[1][3] : 3
A[1][4] : 5
A[2][1] : -6
A[2][2] : 8
A[2][3] : 4
A[2][4] : 7
A[3][1] : 2
A[3][2] : 6
A[3][3] : 9
A[3][4] : -1
The solution is:

x1=-7.809084
x2=-8.690902
x3=7.418177
```

Q. Write a program to implement Gauss Jordan Method.

```
#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);
}
```

OUTPUT

```
/tmp/L57tXf6yYB.o
Enter the size of matrix: 3
Enter the elements of augmented matrix row-wise:
 A[1][1]:10
 A[1][2]:-7
 A[1][3]:5
 A[1][4]:9
 A[2][1]:3
 A[2][2]:6
 A[2][3]:0
 A[2][4]:-9
 A[3][1]:9
 A[3][2]:3
 A[3][3]:-2
 A[3][4]:-1
 The solution is:

 x1=0.224806

 x2=-1.612403

 x3=-0.906976
```

Q. Write a program to implement Newton's Forward Interpolation Formula.

```
#include<stdio.h>
#define MAXN 100
#define ORDER 4

main()
{
   float ax[MAXN+1], ay [MAXN+1], diff[MAXN+1][ORDER+1],
   nr=1.0, dr=1.0,x,p,h,yp;
   int n,i,j,k;
   printf("\nEnter the value of n:\n");
   scanf("%d",&n);

   printf("\nEnter the values in form x,y:\n");
   for (i=0;i<=n;i++)
      scanf("%f %f",&ax[i],&ay[i]);
   printf("\nEnter the value of x for which the value of y is wanted: \n");
   scanf("%f",&x);
   h=ax[1]-ax[0];

   for (i=0;i<=n-1;i++)
      diff[i][1] = ay[i+1]-ay[i];

   for (j=2;j<=ORDER;j++)
      for(i=0;i<=n-j;i++)
      diff[i][j] = diff[i+1][j-1] - diff[i][j-1];
   i=0;
   while (!(ax[i]>x))
      i++;

   i--;
   p = (x-ax[i])/h;
   yp = ay[i];

   for (k=1;k<=ORDER;k++)
   {
      nr *=p-k+1;
      dr *=k;
      yp +=(nr/dr)*diff[i][k];
   }
   printf("\nWhen x = %6.1f, corresponding y = %6.2f\n",x,yp);
}
```

OUTPUT

```
/tmp/kIEEEUQc7C.o
Enter the value of n:
6
Enter the values in form x,y:
100 10.63
150 13.03
200 15.04
250 16.81
300 18.42
350 19.90
400 21.27
Enter the value of x for which the value of y is wanted:
218
When x =  218.0, corresponding y =  15.70
```

Q. Write a program to implement Newton's Backward Interpolation Formula.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
float u_cal(float u, int n)
{
        float temp = u;
        for (int i = 1; i < n; i++)
                temp = temp * (u + i);
        return temp;
}
int fact(int n)
{
        int f = 1;
        for (int i = 2; i <= n; i++)
                f *= i;
        return f;
}

void main()
{
        int n=0;
        printf("Please Enter the number of values: ");
        scanf("%d",&n);
        float x[n],y[n][n];
        printf("Please enter the values of x:\n");
        for(int i=0;i<n;i++){
      scanf("%f",&x[i]);
        }
        printf("Please enter the values of y:\n");
        for(int i=0;i<n;i++){
      scanf("%f",&y[i][0]);
        }


        for (int i = 1; i < n; i++) {
                for (int j = n - 1; j >= i; j--)
                        y[j][i] = y[j][i - 1] - y[j - 1][i - 1];
        }


        printf("Displaying the backward difference table.....\n\n");
```

```
        for (int i = 0; i < n; i++) {
          printf("%0.2f\t\t",x[i]);
                for (int j = 0; j <= i; j++)
                        printf("%0.2f\t\t",y[i][j]);

                printf("\n");
        }


        float value = 0;
        printf("Please enter the value of f\n");
        scanf("%f",&value);

        float sum = y[n - 1][0];
        float u = (value - x[n - 1]) / (x[1] - x[0]);
        for (int i = 1; i < n; i++) {
                sum = sum + (u_cal(u, i) * y[n - 1][i]) /
                                                                fact(i);
        }

        printf("Value at %0.4f is %0.4f\n",value,sum);

}
```

OUTPUT

```
/tmp/zfiJJmnY8n.o
Please Enter the number of values: 5
Please enter the values of x:
1891
1901
1911
1921
1931
Please enter the values of y:
46
66
81
93
101
Displaying the backward difference table.....

1891.00     46.00
1901.00     66.00       20.00
1911.00     81.00       15.00       -5.00
1921.00     93.00       12.00       -3.00       2.00
1931.00     101.00      8.00        -4.00       -1.00       -3.00
Please enter the value of f
1925
Value at 1925.0000 is 96.8368
```

Q. Write a program to implement Lagrange's Interpolation Formula.

```
#include<stdio.h>
main()
{
   float x[100],y[100],a,s=1,t=1,k=0;
   int n,i,j,d=1;
   printf("\n\n Enter the number of the terms of the table: ");
   scanf("%d",&n);
   printf("\n\n Enter the respective values of the variables x and y: \n");
   for(i=0; i<n; i++)
   {
      scanf ("%f",&x[i]);
      scanf("%f",&y[i]);
   }
   printf("\n The table you entered is as follows : \n");
   for(i=0; i<n; i++)
   {
      printf("%0.3f\t%0.3f",x[i],y[i]);
      printf("\n");
   }
   while(d==1)
   {
      printf(" \n Enter the value of the x to find the respective value of y\n");
      scanf("%f",&a);
      for(i=0; i<n; i++)
      {
         s=1;
         t=1;
         for(j=0; j<n; j++)
         {
            if(j!=i)
            {
               s=s*(a-x[j]);
               t=t*(x[i]-x[j]);
            }
         }
         k=k+((s/t)*y[i]);
      }
      printf("\n\n The respective value of the variable y is: %f",k);
   }
}
```

OUTPUT

```
/tmp/zfiJJmnY8n.o
Enter the number of the terms of the table: 5
Enter the respective values of the variables x and y:
5 150
7 392
11 1452
13 2366
17 5202
The table you entered is as follows :
5.000   150.000
7.000   392.000
11.000  1452.000
13.000  2366.000
17.000  5202.000

 Enter the value of the x to find the respective value of y
9
The respective value of the variable y is: 810.000000
```

Q. Write a program to implement Trapezoidal Method.

```
#include<stdio.h>
#include<math.h>
float f(float x)
{
   return(1/(1+pow(x,2)));
}
void main(){
   int i,n;
   float x0,xn,h,y[20],so,se,ans,x[20];
   printf("\n Enter values of x0,xn,h:\n");
   scanf("%f%f%f",&x0,&xn,&h);
   n=(xn-x0)/h;
   if(n%2==1)
   {
      n=n+1;
   }
   h=(xn-x0)/n;
   printf("\nrefined value of n and h are:%d  %f\n",n,h);
   printf("\n Y values \n");
   for(i=0; i<=n; i++)
   {
      x[i]=x0+i*h;
      y[i]=f(x[i]);
      printf("\n%f\n",y[i]);
   }
   so=0;
   se=0;
   for(i=1; i<n; i++)
   {
      if(i%2==1)
      {
         so=so+y[i];
      }
      else
      {
         se=se+y[i];
      }
   }
   ans=h/3*(y[0]+y[n]+4*so+2*se);
   printf("\nfinal integration is %f",ans);
}
```

OUTPUT

```
Enter values of x0,xn,h:
0 3 0.5
refined value of n and h are:6  0.500000

 Y values

1.000000
0.800000
0.500000
0.307692
0.200000
0.137931
0.100000
final integration is 1.247082
```

Q. Write a program to implement Simpson 1/3 rule.

```
#include<stdio.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%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;
}
```

OUTPUT

```
Enter lower limit of integration: 0
Enter upper limit of integration: 1
Enter number of sub intervals: 6
Required value of integration is: 0.785
```

Q. Write a program to implement Simpson 3/8 rule.

```
#include<stdio.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;
}
```

OUTPUT

```
Enter lower limit of integration: 0
Enter upper limit of integration: 1
Enter number of sub intervals: 12
Required value of integration is: 0.785
```

Q. Write a program to implement Euler's method.

```
#include<stdio.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;
 }

 printf("\nValue of y at x = %0.2f is %0.3f",xn, yn);

 return 0;
}
```

OUTPUT

```
Enter Initial Condition
x0 = 0
y0 = 1
Enter calculation point xn = 1
Enter number of steps: 10
x0  y0  slope   yn
------------------------------
0.0000  1.0000  1.0000  1.1000
0.1000  1.1000  1.2000  1.2200
0.2000  1.2200  1.4200  1.3620
0.3000  1.3620  1.6620  1.5282
0.4000  1.5282  1.9282  1.7210
0.5000  1.7210  2.2210  1.9431
0.6000  1.9431  2.5431  2.1974
0.7000  2.1974  2.8974  2.4872
0.8000  2.4872  3.2872  2.8159
0.9000  2.8159  3.7159  3.1875

Value of y at x = 1.00 is 3.187
```

Q. Write a program to implement the Runge Kutta method.

```
#include<stdio.h>
#include<math.h>
float f(float x,float y);
int main()
{
   float x0,y0,m1,m2,m3,m4,m,y,x,h,xn;
   printf("Enter x0,y0,xn,h:");
   scanf("%f %f %f %f",&x0,&y0,&xn,&h);
   x=x0;
   y=y0;
   printf("\n\nX\t\tY\n");
   while(x<xn)
   {
      m1=f(x0,y0);
      m2=f((x0+h/2.0),(y0+m1*h/2.0));
      m3=f((x0+h/2.0),(y0+m2*h/2.0));
      m4=f((x0+h),(y0+m3*h));
      m=((m1+2*m2+2*m3+m4)/6);
      y=y+m*h;
      x=x+h;
      printf("%f\t%f\n",x,y);
   }
}
float f(float x,float y)
{
   float m;
   m=(x-y)/(x+y);
   return m;
}
```

OUTPUT

```
Enter x0,y0,xn,h:0 2 2 0.5
X       Y
0.500000    1.621356
1.000000    1.242713
1.500000    0.864069
2.000000    0.485426
```