Write C code for finding roots

1) Bisection Method

```
#include <stdio.h>
#include <math.h>
 float fun (float x)
{
return (x*x*x' - 4*x - 9);
}
void bisection (float* x, float a, float b, *itr)
{
  x = (a+b)/2;
  ++(*itr);
  printf("Iteration no %3d x = %7 If/n ";itr, *x)
}
void main()
{
  int itr = 0, maxmitr;
  float x, a, b, aller, x1;
  printf("\n Enter value of a, b allowed error
  & maxi. Iter: \n",
  scanf("%f, %f, %f, %d", &a, &b, &aller, &maxmitr)
  bisection (&x, a, b, &itr);
  do {
  If (fun(a) * fun (x) < 0)
      b = x;
  else
      a = x;
  bisection (&x1, a, b &itr);
```

```
If (fabs (x1-x) < allerr)
{ printf (" After %d iterations, root = %.6f, f(%f) = %f", itr, x1)
  return 0;
  }
  x = x1;
  }
  while (itr < maxmitr);
  printf (" The sol" does not converge or iterations
  are not sufficient");
  return 1;
  }
```

2) Regula - Falsi Method

```
# include <stdio.h>
# include <conio.h>
# include <math.h>
  int main()
{ float x0, x1, x2, f0, f1, f2, e;
  int step = 1;
  clrscr();
  printf("\n Enter two initial guesses: \n");
  scanf("%f%f", &x0, &x1);
  printf(" Enter tolerable error \n");
  scanf("%f", &e);
  f0 = f(x0);
  f1 = f(x1);
  if (f0 * f1 > 0.0)
```

```
{ printf (" Incorrect initial guess");
  goto up;
}
printf ("\n step \t\t x0 \t\t x1 \t\t x2 \t\t
        \t f(x2) \n");
do
{ x2 = x0 - (x0 - x1) * f0 / (f0 - f1);
  f2 = f(x2);
  printf ("%. d \t\t %. f \t\t %. f \t\t %. f \n"
    step x0, x1, x2, f2);
  if (f0 * f2 < 0)
  { x1 = x2;
    f1 = f2;
  } else
  { x0 = x2;
    f0 = f2; }
  step = step + 1;
} while (fabs (f2) > e);
printf ("\n Root is : %. f", x2);
  getch();
  return 0;
  }
```

3) Secant Method

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

```
#define f(x) x*x*x-2*x-5
void main()
{ float x0, x1, x2, f0, f1, f2, e;
  int step=1, N;
  clrscr();
  printf("\n Enter initial guesses :\n");
  scanf("%f%f", &x0, &x1);
  printf("Enter tolerable error :\n");
  scanf("%f", &e);
  printf(" Enter max iteration\n");
  scanf("%d", &N);
  printf("\n step\t\t x0\t\t x1\t\t\n");
  do
  { f0 = f(x0);
    f1 = f(x1);
    if (f0==f1);
  { printf("Mathematical error\n");
    exit(0);
  }
    x2 = x1-(x1-x0)*f1/(f1-f0);
    f2 = f(x2);
printf("%d\t\t%f\t\t%f\t\t%f\n",
  step, x0, x1, x2, f2);
  x0 = x1;
  f0 = f1;
  x1 = x2;
  f1 = f2
  step = step+1;
```

```
if (step > N)
{ printf (" Not convergent ");
  exit (0);
}
while (fabs (f2) > e);
printf ("\n root is : %f", x2);
getch();
}
```

4) Fixed point Method

```
# include <stdio.h>
# include <math.h>
float raj (float);
main()
{ float a[100], b[100], c=100.0;
  int i='1', j=0;
  b[0] = (cos(0) - 3* 0 + 1);
  printf ("\n Enter initial guess: \n");
  scanf ("%f", &a[0]);
  printf ("\n\n The value of iterations are: \n");
while (c > 0.00001)
{ a[j+1] = raj [a[j]);
  c = a[j+1] = a[j];
  c = fabs (c);
  printf ("%d %f \n", j, a[j]);
  j++;
}
printf ("\n root of given function is ", a[j]);
}
```

```
float y;
y = (cos(x) + 2)/3;
return y;
}
```

5) Lagranges Interpolation

```
#include <stdio.h>
#include <conio.h>
void main()
{ float x[100], y[100], xp, yp = 0, p;
  int i, j, n;
  clrscr();
  printf("Enter no. 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);
 getch();
}
```

6) Euler's Method

```
#include <iostream>
#define f(x, y) x+y
using namespace std;
int main()
{ float x0, y0, xn, h, yn, slope;
  int i, n;
  cout << "Enter initial condition" << endl;
  cout << "x0 = ";
  cin >> x0;
  cout << "y0 = ";
  cin >> y0;
  cout << "Enter calculation point xn = ";
  cin >> xn
  h = (xn - x0) / n;
  cout << "\n x0 \t y0 \t slope \t yn \n";
  for (i = 0; i < n; i++)
  {
   slope = f(x0, y0);
   yn = y0 + h * slope;
```

```
cout << x0 << "\t" << y0 << "\t" << slope << "\t"
                                << yn << endl;
    y0 = yn
    x0 = x0 + h;
} cout << "\n value of y at x = " << xn << "is"
                                << yn;
  return 0;
}
```

7) Taylor's series -

```
#include <iostream>
#include <math.h>
#include <iomanip>
using namespace std;
int main()
{
  double x0 = 0, y0 = 1, h = 0, y1, y2, y3, y4, y;
  y1 = 3* x0 + y0 * y0;
  y2 = 3 + 2* y0 * y1;
  y3 = 2* y1 * y1 + 2 * y0 * y2;
  y = y0 + (y1 * h) + (y2 * pow(h,2))/2 + ... ;
  cout << "The value of y when x = 0.1 is" << setprecision
                    (5) << fixed << y << end return 0;
}
```

8) Runga kutta method

```
#include <iostream>
# define f(x,y) (y*y - x*x)/(y*y + (x)*x)
```

```
using namespace std;
```

```
using namespace std;
int main()
{ float x0, y0, xn, h, yn, k1, k2, k3, k4, k;
  int i, n;
  cout << "Enter initial condition" << endl;
  cout << "x0 = ";
  cin >> x0;
  cout << "y0 = ";
  cin >> y0
  cout << "Enter calculation point xn= ";
  cin >> xn;
  cout << "Enter no. of step :";
  cin >n;
  h = (xn - x0)/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;
  cout << x0 << " \t " << y0 << " \t " << yn << endl;
    x0 = x0 + h;
    y0 = yn;
  }
   cout << "\n value of y at x = " << xn << " is " << yn;
   return 0;
  }
```