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

double f(double x)
{
	return (exp(-x) - (x*x*x));
}

double dfdx(double x)
{
	return ((-3*(x*x)) - (exp(-x)));
}


double planck_deriv(double x)
{
	double deriv = ((x*exp(x))/(exp(x)-1.0)) - 5.0;
	return deriv;
}


int bisection_f(double a, double b, int nmax, double eps)
{
	int n=0;
	double c, fa, fb, fc;

	fa = f(a);
	fb = f(b);

	if((fa * fb) > 0 ) {		// if the ranges have the same sign
		printf("The range of values [%f, %f] produce values f(%f)=%f and f(%f)=%f
			with the same sign\n", a,b,a,fa,b,fb);
		printf("Choose different range values\n");
		exit(1);
	}
	else {	//the root lies within this range
		for(n = 1; n < nmax; n++)
		{
			c = (a+b)/2.0;
			fc = f(c);

			printf("Iteration no. %d, c=%f, f(c)=%f\n", n,c,fc);

			if(((fabs(b-a)) < eps) || (fc == 0))  {
				printf("Converged at Iteration no. %d | the root = %f\n", n, c);
				return n;
			}
```

```
                        if((fa * fc) < 0) {
                                b = c;
                        }
                        else {
                                a = c;
                        }
                }
                printf("Did not converge. Try a different nmax value\n");
        }
}


int newton_f(double x, int nmax, double eps)
{
        int n = 0;
        double d;

        printf("Iteration no. %d, Initial guess x=%f, f(x)=%f\n", n,x,f(x));
        for(n = 1; n < nmax; n++)
        {
                if(fabs(dfdx(x)) < 0.00001) {    // if the derivative is near zero
                        printf("The derivative is near zero, dfdx(x)=%f\n",dfdx(x));
                        exit(1);
                }

                d = f(x)/dfdx(x);
                x = x - d;
                printf("Iteration no. %d, x=%f, f(x)=%f\n", n,x,f(x));

                if(fabs(d) < eps) {
                        printf("Converged at Iteration no. %d | the root = %f\n", n, x);
                        return n;
                }
        }

        printf("Did not converge. Try a different nmax value\n");
}


int secant_f(double x1, double x2, int nmax, double eps)
{
        int n = 0;
        double d;

        printf("Iteration no. %d, Initial guesses x1=%f, x2=%f, f(x1)=%f, f(x2)=%f\n",
                  n,x1,x2,f(x1),f(x2));
        for(n = 1; n < nmax; n++)
        {
                double dfdx = (f(x2)-f(x1))/(x2-x1);
```

```
		if(fabs(dfdx) < 0.00001) {	// if the derivative is near zero
			printf("The derivative is near zero, dfdx=%f\n",dfdx);
			exit(1);
		}

		d = f(x2)/dfdx;
		double x3 = x2 - d;
		printf("Iteration no. %d, x1=%f, x2=%f, x=%f, f(x)=%f\n",n,x1,x2,x3,f(x3));

		if(fabs(d) < eps) {
			printf("Converged at Iteration no. %d | the root = %f\n", n, x3);
			return n;
		}

		x1 = x2;
		x2 = x3;
	}

	printf("Did not converge. Try a different nmax value\n");
}


int secant_planck(double x1, double x2, int nmax, double eps)
{
 int n = 0;
 double d;

 printf("Iteration no. %d, Initial guesses x1=%f, x2=%f, planck_deriv(x1)=%f,
planck_deriv(x2)=%f\n",
		n,x1,x2,planck_deriv(x1),planck_deriv(x2));
 for(n = 1; n < nmax; n++)
  {
   double dfdx = (planck_deriv(x2)-planck_deriv(x1))/(x2-x1);

   if(fabs(dfdx) < 0.00001) {	// if the derivative is near zero
		printf("The derivative is near zero, dfdx=%f\n",dfdx);
		exit(1);
   }

   d = planck_deriv(x2)/dfdx;
   double x3 = x2 - d;
   printf("Iteration no. %d, x1=%f, x2=%f, x=%f, planck_deriv(x)=%f\n",
	n,x1,x2,x3,planck_deriv(x3));

   if(fabs(d) < eps) {
		printf("Converged at Iteration no. %d | the root = %f\n", n, x3);
		return n;
   }

   x1 = x2;
```

```
    x2 = x3;
   }

  printf("Did not converge. Try a different nmax value\n");
}

int main(int argc, char* argv[])
{
        printf("PROBLEM 1: STARTING BISECTION METHOD\n");
        printf("Total \"steps\" = %d\n\n",bisection_f(-10.0, 10.0, 30, 0.000001));

        printf("PROBLEM 1: STARTING NEWTON'S METHOD\n");
        printf("Total \"steps\" = %d\n",newton_f(10.0, 20, 0.000001));

        printf("PROBLEM 1: STARTING SECANT METHOD\n");
        printf("Total \"steps\" = %d\n",secant_f(9.9,10.0, 20, 0.000001));

        printf("PROBLEM 2: STARTING SECANT METHOD ON PLANCK DERIVATIVE\n");
        printf("Total \"steps\" = %d\n",secant_planck(1.0,2.0, 20, 0.000001));
        return 0;
}
```