```c
/**
 * @author  Aviruk Basak, CSE214047, Sem 3, Year 2
 * @topic   Using Bisection Method to find f(a) for a given data set and a given value of 'a'
 * @date    1-8-2022
 * @cc      gcc -Wall -lm -o bisection-method bisection-method.c
 */

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

# define TOLERANCE      (0.001)
# define MAX_ITERATIONS (1000)
# define FLOAT_FORMAT   "%0.3lf"

typedef struct {
    double a;
    double b;
} Tuple;

int signum(double x);
void printRoot(Tuple intrvl);
Tuple bisectAndSolve(double (*f)(double x), Tuple intrvl);

int signum(double x)
{
    if (x < 0)
        return -1;
    else if (x > 0)
        return +1;
    else
        return x;
}

void printRoot(Tuple intrvl)
{
    double ea, er, ep;
    ea = intrvl.a - intrvl.b;
    er = ea / intrvl.a;
    ep = er * 100;
    printf("  result = " FLOAT_FORMAT ", " FLOAT_FORMAT "\n", intrvl.a, intrvl.b);
    printf("    exact error      = " FLOAT_FORMAT "\n", ea);
    printf("    relative error   = " FLOAT_FORMAT "\n", er);
    printf("    percentage error = " FLOAT_FORMAT "\n", ep);
}

Tuple bisectAndSolve(double (*f)(double x), Tuple intrvl)
{
    double a, b, t, fa, fb, ft;
    size_t i = 0;
    a = intrvl.a;
    b = intrvl.b;
    printf("\t a\t b\t t\t f(a)\t f(b)\t f(t)\n");
    while (i < MAX_ITERATIONS) {
        fa = f(a);
        fb = f(b);
        t = (a + b) / 2;
        ft = f(t);
        printf("i:%zu:\t " FLOAT_FORMAT "\t " FLOAT_FORMAT "\t " FLOAT_FORMAT "\t " FLOAT_FORMAT "\t " FLOAT_FORMAT "\t " FLOAT_FORMAT "\n", i, a, b, t, fa, fb, ft);
        if (f(t) == 0 || fabs(a - b) < TOLERANCE) {
            intrvl.a = a;
            intrvl.b = b;
            return intrvl;
        }
        if (signum(fa) == signum(ft)) {
            a = t;
        } else if (signum(fb) == signum(ft)) {
            b = t;
        }
        i++;
    }
    intrvl.a = a;
    intrvl.b = b;
    return intrvl;
}

double f1(double x)
{
    // f(x) = x⁴ - x - 10
    return pow(x, 4) - x - 10;
}

double f2(double x)
{
    // f(x) = x - e^(-x)
    return x - exp(-x);
}

double f3(double x)
{
    // f(x) = e^(-x) - 3 log(x)
    return exp(-x) - 3 * log(x);
}

double f4(double x)
{
    // f(x) = e^(-x) * (x² + 5x + 2) + 1
    return exp(-x) * (pow(x, 2) + 5 * x + 2) + 1;
}

int main()
{
    Tuple it1 = { 1, 2 };
    printf("\nf(x) = x⁴ - x - 10\n");
    printRoot(bisectAndSolve(f1, it1));

    Tuple it2 = { 0, 1 };
    printf("\nf(x) = x - e^(-x)\n");
    printRoot(bisectAndSolve(f2, it2));

    Tuple it3 = { 1, 1.368 };
    printf("\nf(x) = e^(-x) - 3 log(x)\n");
    printRoot(bisectAndSolve(f3, it3));

    Tuple it4 = { 1, 2 };
```

```
117     printf("\nf(x) = e^(-x) * (x² + 5x + 2) + 1\n");
118     printRoot(bisectAndSolve(f4, it4));
119
120     return 0;
121 }
```