Homework Assignment 2
Due Friday, September 8 at 6 pm
(Note change in due date)

Write a complete C program that implements the bisection method for finding a solution to the equation \( f(x) = 0 \). Input to the program will be an initial “left guess” and “right guess” (see below), the desired maximum error tolerance, and the maximum number of iterations. Output should be:

- The maximum number of iterations and the actual number of iterations,
- The maximum error tolerance and an estimate of the actual error,
- The approximate solution and the value of the function \( f(x) \) at the approximate solution.

You should assume that the input is valid: you don’t need to check it for correctness.

**The Bisection Method**

The bisection method is an algorithm for finding approximate solutions to \( f(x) = 0 \). The method begins with two initial guesses, \( a \) and \( b \), that “bracket” the (unknown) solution. That is, \( a \) is less than the solution and \( b \) is greater than the solution. For this assignment you can assume that \( a \) and \( b \) have been chosen so that \( f(a) < 0 \) and \( f(b) > 0 \), and there’s only one solution to \( f(x) = 0 \) between \( a \) and \( b \). So you can assume that there must be a point \( c, a < c < b \), such that \( f(c) = 0 \).

Once we know \( a \) and \( b \), we can search for the zero using an approach that’s very similar to binary search:

- Find the midpoint:
  \[
  m = \frac{a + b}{2}
  \]

- If \( f(m) = 0 \), we’re done!
- If \( f(m) < 0 \), replace \( a \) by \( m \).
- If \( f(m) > 0 \), replace \( b \) by \( m \).
This insures that we always have $f(a) < 0$ and $f(b) > 0$. Now we repeat this procedure. At each stage we halve the length of the interval containing the solution.

In general methods like the bisection method won’t find the exact solution $c$: they’ll only find an approximate solution — in our case $m$. So we input a maximum error tolerance; call it $e$. Since the interval from $a$ to $b$ should contain the solution $c$, $m$ should be at least as close to $c$ as the length of the interval from $a$ to $b$. (Draw a picture!) So when $b - a \leq e$, we can stop.

As we’ve described it, the bisection method assumes that the initial guesses $a$ and $b$ will satisfy $f(a) < 0$ and $f(b) > 0$. This is to insure that there is a point $c$ with $a < c < b$ such that $f(c) = 0$. In general, we only need one of $f(a)$ and $f(b)$ to be negative and the other positive. It’s also not necessary to assume that there is only one solution to $f(x) = 0$ between $a$ and $b$, but these assumptions make our code quite simple and easy to understand.

So we have the following pseudocode:

```plaintext
Input: a, b, tol, max_iters [3 doubles and an int]

/* f(a)*f(b) < 0 */
iters = 0;
while (iters < max_iters && fabs(b-a) > tol) {
    iters++;
    mid = (a + b)/2;
    /* Do we replace a or b? */
    if (f(mid) == 0) {
        done; break; /* This will exit the while loop */
    } else if (f(mid) < 0) {
        a = mid;
    } else /* f(mid) > 0 */ {
        b = mid;
    }
}

print solution = mid, etc;
```

The Program
As noted above, the input will be

```
a, b, tol, max_iters
```
and you can assume that $f(a) < 0$ and $f(b) > 0$. Also as noted above, output should include
• The maximum number of iterations, the actual number of iterations

• The maximum acceptable tolerance, the actual difference $b - a$ between the final values of $b$ and $a$.

• The approximate solution and the function value at the approximate solution

Doubles should be output in exponential notation with 10 places to the right of the decimal. So use the format specifier \%.10e.

You must use a C function to implement the bisection method. It must have the following properties:

• The return value is the approximate solution

• It takes four input args: $a$, $b$, tol, max_iters

You can use more arguments if there’s a good reason for them.

Note that the function $f(x)$ should be hardwired in the source code: it should not be input by the user. Here are some functions and data you can test your code with.

• $f(x) = x^2 - 2$, $a = 1$, $b = 2$, $c = \sqrt{2}$.

• $f(x) = (x - 1)^3$, $a = 0.5$, $b = 2$, $c = 1$.

• $f(x) = 4x - 3$, $a = 0.0$, $b = 2$, $c = 0.75$.

When you submit the final version of your program, your source code should have the function $f(x) = x^2 - 2$.

Typing in the tolerance will probably be easier if you use exponential notation: for example, it’s probably easier to type $1.0e-6$ than 0.000001.

**Submission** Create a subdirectory h2 of your submit directory and copy your source code to this directory:

```bash
$ scp my_h2.c <my_userid>@stargate.cs.usfca.edu:
$ ssh <my_userid>@stargate.cs.usfca.edu
$ mkdir /home/submit/<my_userid>/h1
$ cp my_h2.c /home/submit/<my_userid>/h1
```