Burak Yesil
“I pledge my honor that I have abided by the Stevens Honor System.”

I first started off by looking at the bisection method algorithm pseudo code on wikipedia. After further examining the pseudocode, I came to the conclusion that all I need is a for loop, a function calculator, and some basic conditional statements.

```
INPUT: Function f,
       endpoint values a, b,
       tolerance TOL,
       maximum iterations NMAX
CONDITIONS: a < b,
            either f(a) < 0 and f(b) > 0 or f(a) > 0 and f(b) < 0
OUTPUT: value which differs from a root of f(x) = 0 by less than TOL

N ← 1
while N ≤ NMAX do // limit iterations to prevent infinite loop
    c ← (a + b)/2 // new midpoint
    if f(c) = 0 or (b − a)/2 < TOL then // solution found
        Output(c)
        Stop
    end if
    N ← N + 1 // increment step counter
    if sign(f(c)) = sign(f(a)) then a ← c else b ← c // new interval
end while
Output("Method failed.") // max number of steps exceeded
```

Consequently, I decided to add two procedures, bisection and function. These two are both the main procedures, but they have inner procedures that are called inside of them.

```
bisection: //returns the final resul at d0

    //Gets addresses of a and b
    adr x1, a
    adr x2, b

    //Storing values into floating point registers
    ldur x0, [x1]
    fmov d1, x0 // d1 = a
    ldur x0, [x2]
    fmov d2, x0 //d2 = b

    //storing tol in the d3 register
    adr x6, tol
    ldur d3, [x6] //d3 = tol

    //storing the n variable in x3
    adr x6, n
    ldr x3, [x6] //x3 = n

    //storing the coeff address in x7
    adr x7, coeff
```

This is the first part of the bisection function. In this section, I am loading most of the data variables from the data section.

```
    //L1 replciates the while loop from the wikipedia pseudocode
    L1:
        fadd d0, d1, d2 //c
        fmov d5, 2.0
        fdiv d0, d0, d5
        fmov d12, d0
        bl function //f(c)

        scvtf d6, xzr //changes int into a double/floating point variable
        fcmp d0, d6 //if f(c) == 0
        b.eq found //exit
        fsub d4, d2, d1
        fmov d11, 2.0
        fdiv d4, d4, d11
        fcmp d4, d3
        b.lt found
        fmov d4, d0 //mov c to d4
        fmov d0, d1
        bl function //d0=f(a)
        fmul d0, d0, d4
        fcmp d0, 0.0
        b.ge same_sign
        b.lt opposite_sign
```

In this section of the document, we are actually making the outer for loop, which represents the while loop in the pseudocode. Moreover, inside of the L1, you can see that we need to call the function procedure in order to actually get the function results that will be used inside of the loop.

```
same_sign:
        fmov d1, d12 //a = c
        b L1

opposite_sign:
        fmov d2, d12 //b=c
        b L1

not_found:
        fmov d0, d0
        adr x0, no_value_msg
        bl printf
        b Exit

found:
        fmov d0, d12
        adr x0, msg
        bl printf
        b Exit
```

These are the extra smaller procedures needed for a few cases. For example, same_sign runs when the two function results have the same sign vs opposite_sign runs when the two function results have opposite signs. Similarly, when an actual final algorithm result is found, we call found, if this isn't the case we call not_found.

```
//exits the program
Exit:
    mov x0, #0
    mov w8, #93
    svc 0


.data
coeff: .double 0.2, 3.1, -0.3, 1.9, 0.2
n: .dword 4 //NMAX = 4
a: .double -1.0 //a = -1
b: .double 1.0 //b = 1
tol: .double 0.01 // tol = .01
msg: .ascii "%lf\n\0"
no_value_msg: .ascii "A value wasn't found\n\0."
.end
```

This is the bottom section of my code. Similar to all of the other programming assignments, I have a data section that has the variables I use throughout my code. I decided the variables I used by looking at the variables used in the wikipedia pseudocode.

Overall, this assignment wasn't too bad because it was mainly just looking at the wikipedia page and