**BIL 105E – Introduction to Scientific and Engineering Computing (C)**

**Spring 2016-2017**

**Homework 3**
**Root Determining With Bisection**

Assignment Date: 20.03.2017
Due Date: 02.04.2017 - 23:59
Duration 2 weeks

In this homework you will determine real roots of polynomials. There are several techniques to determine real roots of polynomials and in this homework you will use bisection method.

## Mathematical Background

Bisection is an iterative method to determine real roots of polynomials (f(x)). Basically, it starts with two points (a,b) from the domain that guarantees the following conditions: 1. One of the points (a) should be smaller than the root, whereas the other (b) should be larger than the root. 2. f(a) and f(b) should have opposite signs.

The bisection method narrows down the interval by recursively selecting a new point (p), and replacing one of the points (a OR b) that holds the conditions defined above (1 and 2). After several iterations, a root is determined with a specified precision.

Bisection finds x where f(x)=0 in the interval [a,b]. To use bisection, the following has to be true:

$$f(a)f(b) < 0$$

If this condition holds, f(a) is negative, while f(b) is positive or vise versa. Since f is a continuous function, there must be a root for f in the interval [a,b].

The bisection method calculates p, the mid point of a and b and replaces p with one of the points (a OR b):

$$\text{If } f(a)f(p) \text{ positive then } a \leftarrow p$$
$$\text{negative then } b \leftarrow p$$

The method should be implemented in a RECURSIVE fashion. It recursively applies the operation above until f(p) is smaller than the error rate ($10^{-3}$).

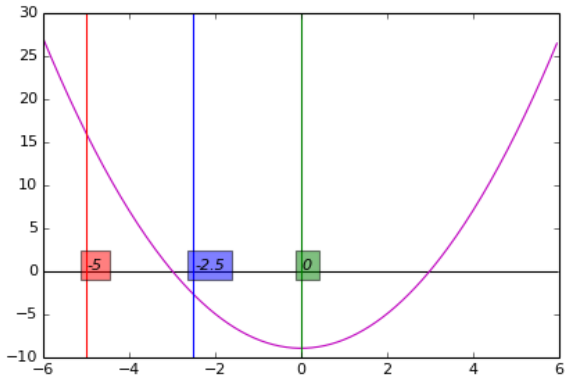
When the bisection method terminates, p should be returned as the root of f(x).

Bisection cannot find double roots (two roots at the same point) since f(a)f(b) < 0 can never be true. You are not required to determine double roots in this homework.

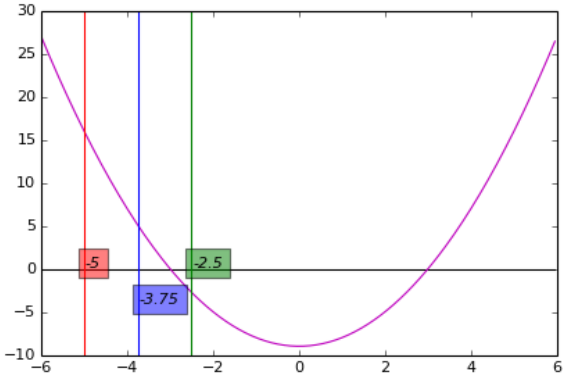
**Example:**

*f(x)=$x^2$ - 9  domain=[-6,6]*

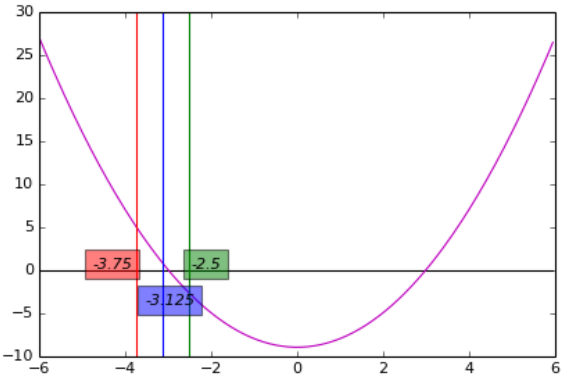
*do bisection between -5,0*



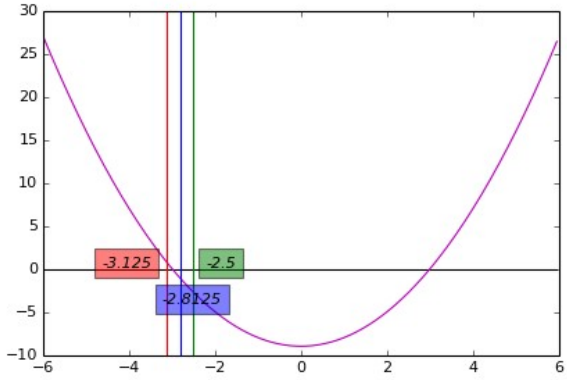

*iteration 1: a = -5, b = 0, p =-2.5*




*iteration 2: a = -5, b = -2.5, p =-3.75*




*iteration 3: a = -3.75, b = -2.5, p =-3.125*




*iteration 4:  a = -3.125, b = -2.5, p =-2.8125*

Graphs above show the first four iteration of the bisection method. It starts the search between -5 and 0. When bisection is done, it has been found that there is a root at x=-3.

Since this is an second degree polynomial, bisection should be run to determine the other root of the polynomial.

*do bisection between 0,5*

The other root would be found as x = 3.

**Important:** You are required to determine the starting parameters of the bisection. In this example I have found f(-5)f(0) < 0 and hence, called the recursive bisection method with a = -5 and b = 0. You should find a clever way to determine higher and lower limits of the bisection. Your code will not be tested with polynomials whose roots have a difference smaller than 1. (Polynomials with roots x=3,x=3.5 will never be the input of this program)

**Compiling:** Your code should be able to compiled using gcc without any errors or warning. Your code will be compiled with the following line while grading.

*gcc homework3.c -std=c99 -o homework3*

Your code will be run as following line while grading.

*./homework3*

**Run Time:** In the run time, user will enter 5 parameters. First three parameters will become the constants of the polynomial functions.
Next 2 parameters will be the search limits. These parameters will limit the domain. Roots of the polynomials will always be in the domain while testing your code. (User will never enter -5 and 5 as limits if a root is at 8.)

***Input Examples***

*Inputs: 1 -4 -5 -50 50 → $x^2$-4x-5 , domain[-50,50]*
*Inputs: 1 0 -5 -50 50 → $x^2$-5 , domain[-50,50]*
*Inputs: 0 -4 -5 -50 50 → -4x-5 , domain[-50,50]*
*Inputs: 0 0 -5 -50 50 → -5 , domain[-50,50]*

The examples on the top are not outputs. They only shows what inputs mean mathematically.

**Implementation Details:** You are expected to implement at least 3 functions.

- equation(...) : This function will work like a mathematical function. It will get the parameters of the polynomial and a point x, then return f(x).
- findRoots(...) : This is the function where root search starts. Your main function calls findRoots(...) function to start finding roots of the polynomial. It prints all roots found. (Prints two roots if there are two roots, one root if there is one root, no root if there are no roots.)
- bisection(...) : This is the function where bisection is done. It works recursively until it detects the roots if they exist.

You are free to implement any other functions if you think you require.

**Output:** After the inputs are entered by the user, your program should print the polynomial and the domain to the console as shown in the example(Do not change the words and order). All the polynomials should be written in a second-degree form regardless of the values of constants. Notice that the negative constants also do not change the polynomial formula as well. Please check the following examples. Following examples are from separate runs.

*Entered equation: 1x^2 + -4x + -5*
*Domain: -50, 50*
*–*
*Entered equation: 0x^2 + -4x + -5*
*Domain: -50, 50*
*–*
*Entered equation: -3x^2 + 4x + 0*
*Domain: -50, 50*

At the termination of the bisection, you should give your outputs as following examples.

If no root could be found, program would print the following line:

*No roots found*

If one root could be found (Line case), program would print the following line:

*Root: x=-1.250*

If two root could be found (Second-degree polynomial case), program would print the following line:

*Roots: x=-5.000, x=1.000*

Your outputs should have 3 digits after decimal point.

**Note1:** You are only interested with real roots.

**Note2:** Second degree polynomials may have double roots(two roots at the same point). This kind of roots can not be detected with bisection. You should omit double roots.(ex: Assume $x^2 + 4x + 4$ has no root)

**Important:** Do not print something else than what is required and requested.

**Termination:** Your code should terminate after it prints the root or the roots. Successful functions return 0. Do not forget to return 0 if your code worked as it should.
(No root is also an successful termination for this program)

**Example Compilation and Run:**

```
airlab@mia:~/Cihan/BIL -105E/Homework3$ gcc homework3.c -std=c99 -o homework3
airlab@mia:~/Cihan/BIL -105E/Homework3$ ./homework3
1 -4 -5 -50 50
Entered equation: 1x^2 + -4x + -5
Domain: -50, 50
Roots: x=-1.000, x=5.000
airlab@mia:~/Cihan/BIL -105E/Homework3$
```

**Evaluation Criterion:**

- Compiling and linking with gcc
- Finding real roots for second degree polynomials if exist in the domain
- Finding real root for first degree polynomials(line) if exist in the domain
- Recursive function bisection
- Successful execution and correct output in requested format

**The homeworks are individual assignments and you are expected to do it by yourself. Any form of cheating will not be tolerated.**

For questions you can contact:
Abdullah Cihan AK
akab@itu.edu.tr