## ECE 2331 - Programming Assignment 1 <br> Finding Roots of Equations

This assignment is designed to teach you how to solve two of the most common problems in numerical methods: (1) evaluating polynomial functions and (2) finding roots of those functions.

## The problem

Your application will take a series of command-line arguments $\left[c_{n}, c_{n-1}, c_{n-2}, \cdots, c_{1}, c_{0}, a, b\right]$ as floating point values. The values $c_{0}$ through $c_{n}$ will be the coefficients of a degree $n$ polynomial $p(x)$. The coefficients are specified in the order of decreasing power. For example, $c_{0}$ is a constant and $c_{n}$ is the coefficient of $x^{n}$. The last two values, $a$ and $b$, will specify the interval over which your application will search for a root and return the following:
a) The location $x=r_{0}$ of the root according to the bisection method
b) The number of iterations required to locate the root using the bisection method
c) The location $x=r_{1}$ of the root according to Newton's method
d) The number of iterations required to locate the root using Newton's method
e) The value of $f(a), f(b), f\left(r_{0}\right), f\left(r_{1}\right)$, and $f^{\prime}\left(r_{1}\right)$

Your code will operate under the following constraints:

1) Return the root position $(x)$ for both methods to an accuracy of $10^{-6}$
2) For Newton's method, start with $x_{0}=\frac{a+b}{2}$
$3)$ Return an error if no root is found within the interval $[a, b]$

Structure your code so that you have three independent functions for calculating:

- the bisection method
- Newton's method
- Horner's algorithm

When writing a function for Horner's algorithm, remember that you'll need to make it flexible enough to solve both the value of the polynomial and its derivative, when necessary.

## Turning in your code

Your code must be written in C or C++ using a single file. Upload your code to Blackboard. I will download it and use a script to test your code out for several examples, so don't do anything weird with the output. Output the values in the order (a through e) described above as standard console output (use std: : cout or printf). You can label the values with text, just don't add any other numbers or change their order. Also, please don't collect any data via the console (everything must be passed via command line parameters).

I will be compiling and grading the code using Visual Studio 2012 (VC++11.0). This is both the industrial standard and consistent with the rest of the department. Visual Studio 2012 is available in the campus computer labs and through Microsoft Dreamspark, where students can obtain the full (professional) version. I certainly don't mind if you write the code on another platform, but make sure to test it using Visual Studio 2012.

## Examples

$x^{3}-6 x^{2}+4 x+12=0 \quad[1,4] \quad x_{r}=2.5173 \quad$ args: 1-641214
$x^{3}-6 x^{2}+4 x+12=0 \quad[4,6] \quad x_{r}=4.53407 \quad$ args: 1-641246
$x^{6}-14 x^{4}+49 x^{2}-36=0 \quad[1.5,2.5]$
$x_{r}=2$
args: 10-140490-361.5 2.5
$x^{7}-14 x^{5}+49 x^{3}-36 x=0 \quad[-4,-2.5]$
$x_{r}=-3$
args: 10-140490-360-4-2.5

Finding Roots of Equations

## Rubric

Name $\qquad$

## input format

## output format

## Horner's algorithm

appropriate precision
code is correct / robust
input and code commented

Bisection method
correct number of iterations
correct root
fails robustly
input and code commented

Newton's Method
correct number of iterations
correct root
input and code commented
correct number of iterations
correct root
input and code commented
correct number of iterations
correct root
input and code commented
—
/ 15
/ 5
/ 5
$\qquad$
/ 5
$\qquad$ / 10
_ $/ 5$
/ 5
_ / 10

Total

