Program Submission Instructions:

- You must submit your source code file
- The source code file must be submitted in Webcourses from the assignment page
- All source code must be in exactly one file of type .c, .cpp, or .java

COT 4500 – Numerical Calculus
Spring 2014
Program #1 Polynomial Root Finder (100 points)

In this assignment you will write a program that will retrieve a simple polynomial, two initial values, and associated parameters from command line arguments specified below, and then run the Newton and Secant Methods to attempt to find a root within the required accuracy and within the maximum number of iterations. The program must generate output to the console (terminal) screen as specified below.

Command Line Parameters

1. Your program compile and run from the command line.
2. The program executable must be named "newton" (all lower case, no spaces or file extension).
3. The program must retrieve the following command line parameters, which must be specified in the order shown below. Your program may NOT prompt the user to enter any parameter values.

Parameter 1:	p0	an integer or decimal real value, can be negative; represents the initial value for Newton's Method and the first initial value for the Secant Method.
Parameter 2:	p1	an integer or decimal real value, can be negative; represents the second initial value for the Secant Method.
Parameter 3:	tol	a decimal real value, must be positive; represents the accuracy tolerance value (e.g., .0005).
Parameter 4:	max	a positive integer; represents the maximum number of iterations that each method will be allowed to run.

Parameter 5:	lim	a positive integer; represents the highest order term of the input polynomial (e.g., lim=3 for a cubic polynomial).
Parameters 6 to (6+lim)		decimal real values for the coefficients of the terms of the polynomial, in descending order of the powers of x, from x^lim to x^0.

Specifying the Polynomial Using Input Parameters

To simplify input of the polynomial, for this assignment the polynomial will be specified by entering its coefficients as decimal real values in descending order of the powers of x, from x^lim to x^0, where the following conditions must be observed:

(a) An implicit coefficient of "1" must be included in the parameter set; and

(b) All terms of the polynomial must be specified, so that terms that do not appear in the polynomial must be represented by zero coefficients in the parameter set.

For example, suppose we wish to enter the function $\mathbf{x^3 - 2x^2 - 5}$. For this function, the x^1 term is not used. Also, this function has an implicit coefficient of 1 for the x^3 term. The parameter set for this function will be: **1 -2 0 -5** .

Please note that the number of parameters needed to specify the polynomial will vary and will always be one more than the degree of the polynomial. For the third order polynomial above, we have four coefficients. For a 4^{th} degree polynomial, 5 coefficients are needed, etc.

Running the Program

To run the program, simply launch the program from a command prompt or terminal window and enter all of the required parameters, in the order specified above, following the name of the program.

For a Java program in Windows, Mac OS, or Linux, an example would be:

java newton 2 4 .0005 50 3 1 -2 0 -5

For a C/C++ version of the program in Mac OS or Linux, the run command would look like

./newton 2 4 .0005 50 3 1 -2 0 -5

And the C/C++ version in Windows would be simply:

newton 2 4 .0005 50 3 1 -2 0 -5

Program Output

The program must send all output to the console (terminal) screen. The output must contain the following sections and must be formatted in substantially the same form as the sample output in the next section:

(a) Echo control parameters: The program must echo the input parameters for p0, p1, tol, max, and lim.

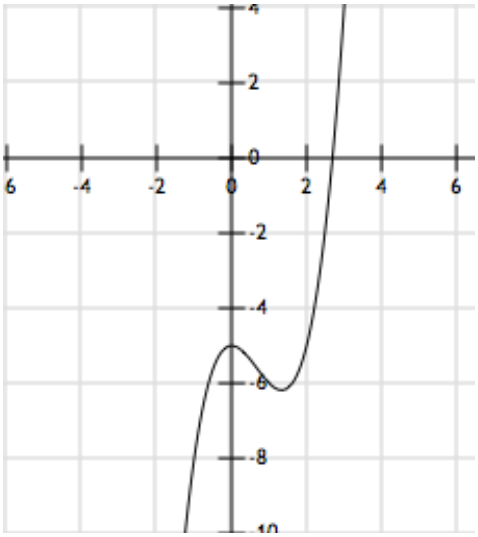
(b) Echo polynomial: The program must echo the polynomial coefficients in a manner that associates each coefficient with the power of x that it precedes. Additionally, negative values should be expressed as such. The x^0 term may be displayed with or without the "x^0".

(c) Report Newton Method Results: The program must show the values of each successive approximation computed in accordance with Newton's Method. The program must also report whether or not a solution was found, and if so after how many iterations. The program must also detect a possible divide by zero situation and exit gracefully with an appropriate message.

(d) Report Secant Method Results: The program must show the values of each successive approximation computed in accordance with the Secant Method. The program must also report whether or not a solution was found, and if so after how many iterations. The program must also detect a possible divide by zero situation and exit gracefully with an appropriate message.

Sample Output

The following is the desired output from running the program using the following command line parameters: 2 4 .0005 50 3 1 -2 0 -5. The function for this output is **$f(x) = x^3 - 2x^2 - 5$.** The graph of this function looks like:



from which we can see that the function has a root somewhere between 2 and 3.

The program output for the above parameter set is:

```
Input Parameters:

      p0  = 2.0
      p1  = 4.0
      tol = 5.0E-4
      max = 50

Polynomial is of order: 3

Terms of polynomial:  1.0*x^3 + -2.0*x^2 + 0.0*x^1 + -5.0*x^0

Newton's Method:

      p1 = 3.25
      p2 = 2.811036789297659
      p3 = 2.697989502468529
      p4 = 2.6906771528603617
      p5 = 2.690647448517619

      Solution found after 5 iterations: 2.690647448517619

Secant Method:

      p2 = 2.3125
      p3 = 2.4977178874157793
      p4 = 2.7424856229400096
      p5 = 2.684791691172416
      p6 = 2.6904821923946063
      p7 = 2.690647985589758

      Solution found after 7 iterations: 2.690647985589758
```

Please note how the terms of the polynomial are shown. This is the recommended display format.

Program Testing

Your program will be tested by running a number of test cases. Some of the test cases will use the same function, but will vary the other parameters. Different functions will also be tested. Scenarios that should be trapped for divide-by-zero errors by both methods will also be tested.

Grading Rubric

The total possible score for this program is 100 points. The following point values will be deducted for the reasons stated:

[-100 points] Your program does not successfully compile from the command line with one of these commands:

C program:	prompt>	gcc –o newton [your_file_name].c
C++ program:	prompt>	g++ –o newton [your_file_name].cpp
Java program:	prompt>	javac newton.java

Note: If you are submitting a Java program, the class file must be named "newton.java" and the class name must be "newton".

[-90 points] Your program does not run from the command line without error or produces no output.

[-70 points] The program compiles, runs, and outputs p0, p1, tol, max, and lim correctly, but crashes thereafter or does not produce any correct output thereafter.

[-50 points] The program compiles, runs, echoes p0, p1, tol, max, and lim correctly, and also reports the polynomial coefficients in the required format.

[-25 points] The program compiles, runs, echoes p0, p1, tol, max, and lim correctly, reports the polynomial coefficients in the required format, and reports correct results for one of the two methods, but not for both.

[no deductions] The program compiles, runs, echoes p0, p1, tol, max, and lim correctly, reports the polynomial coefficients in the required format, and reports correct results for both of the methods.