Homework #6 - C Module for Real Polynomials

Objective: Implement a C99 module for real polynomials that uses Laguerre's method to find all the roots. Become familiar with the complex number arithmetic using **<complex.h>**. Use your **ClassError.h** file from HW 4/5.

The problem

The code framework **poly.c** (in **hw6_files**) describes the public interface for the module functions that you will implement. The most important function is **roots** that finds all the roots of arbitrary real polynomials using Laguerre's method. Your implementation should take advantage "known" root finding short cuts:

1) If a resulting polynomial is order 2 then you should use the quadratic equation.
2) If a root is complex you should automatically use the conjugate.
3) If the resulting polynomial is of order 1 you should calculate the root using simple math.

Since polynomials have in general complex roots you will use the **complex.h** C99 library. Your implementation should provide safety mechanisms to check preconditions, avoid infinite number of iterations, inconsistent data, etc.

Use the header **poly.c** and **poly.h** to guide your development. Note that you must use the following typemark for polynomials.

```
typedef struct { unsigned int        nterms;        /* number of terms */
                 double complex *polyCoef;        /* polynomial coeffs */
               } polynomial;
```

Notes:

- You must use **–lm** on your link line to include the C99 math runtime libraries.
- You are not allowed to use the cpow() function when evaluating roots, you must use Horner's factorization.
- You must create two private functions: **laguerre** and **deflate** in **poly.c.** Private functions are function whose scope is local to the module only so are declared **static**. These functions are not specified in **poly.h** since they would never be available to the user of the module.

Testing and Specifications

Write a "driver" program called **hw6.c** to test your module. The driver program should write errors to **stderr,** results to **stdout,** must use getopt_long_only().and satisfy the following specifications:

a) Support the following command line syntax:

hw6 -input file <-verbose>

With the following abbreviations:

-input -in, -i
-verbose -verb, -v

b) The program should print a friendly usage message if called without the correct number of arguments. The friendly message should include a brief description of the program, the syntax and the meaning of key options.

c) The optional verbose command must include the following messages when appropriate:

Found final two roots through quadratic formula

Found final root with simple math

Found root <root>

Found imaginary root, deflating twice

Deflated <polynomial>

Laguerre intermediate terms x, G(x), H(x), and Alpha.

d) The following examples illustrates how the driver program should be designed

- To find the roots of some polynomials you will type

./hw6 -input polynomials.txt

The results will be printed to **stdout**. For example if the polynomial was $p(x) = x^3+x+1$ the screen should display

```
P(x) =  1x^3 + 1x^1 + 1x^0
Roots:
         0.341164  + -1.16154i
         0.341164  + 1.16154i
         -0.682328
```

./hw6 -in polynomials.txt -verb

```
P(x) =  1x^3 + 1x^1 + 1x^0
        Laguerre's Algorithm( tol = 1e-09 )
          it: 0 x: 0.0
                G(x): 1
            H(x): 1
            Alpha: 1
          it: 1 x: -1
                G(x): -4
            H(x): 10
            Alpha: -0.322876
          it: 2 x: -0.677124
                G(x): 191.326
            H(x): 36933
            Alpha: 0.00520349
          it: 3 x: -0.682328
                G(x): -3.00183e+07
            H(x): 9.01098e+14
            Alpha: -3.3313e-08
        Found root -0.682328 + 0i
        Deflated: P(x) =  1x^2 - 0.682328x^1 + 1.46557x^0
        Found final two roots through quadratic formula
Roots:
         0.341164  + -1.16154i
         0.341164  + 1.16154i
         -0.682328
```

Relative indenting is important.

e) The input (data) files should satisfy the following specifications:

i) For the polynomials it should contain the coefficients of the polynomials, one polynomial per line.

For example, the polynomials

$p_1(x) = x^5+2x-1$

$p_2(x) = 5x^4-4x^3+3x^2-2x-1$

Will be entered as follows:

1.0 0.0 0.0 0.0 2.0 -1.0

5.0 -4.0 3.0 -2.0 -1.0

where the polynomial coefficients are separated by spaces. Note also that "missing powers" in the polynomial **must** be represented by zero coefficients and all the coefficients must be real numbers.

Makefile:

You must provide a quality Makefile with the following targets: all, roots, mem, help and clean.

"all"	-should make **hw6.**
"roots"	- should run **hw6** with **polynomial.txt,** redirected to **out.txt.**
"mem"	**-** should run the roots test using **valgrind** redirected to **mem.txt**
help, clean	- should do the normal things

Modules should be compiled independently of the driver program, that is, you should be able to generate an object file called **poly.o** that can be linked to any program that uses the module functions. Your makefile must include a VERBOSE makefile variable, set to "" that will be used to enabled/disable verbose mode in your code. I will set your VERBOSE makefile variable to "-verb" if I want to see your detailed output.

Implementation Hints

- The file "simple.txt" is included and contains some simple test cases.

Simple.txt	Equation	Roots
1 1	x + 1	-1
1 0 1	x**2 + 1	0.0 + 1.0i, 0.0 -1.0i
1 2 1 2	x**3 + 2x**2 + x + 2	-2.0, 0.0 + 1.0i, 0.0 -1.0i,

Results and Analysis

- Run the test with the polynomials in the file **polynomials.txt** (with -**verbose** to print intermediate results) and redirect the output to a **out.txt** file. Include the results of out.txt in your analysis.txt file.
- Re run the test with Valgrind and redirect the output to a **mem.txt** file.
- Write in the file **analysis.txt** a short explanation of the implementation and organization of the module.
- Prepare a tarball **lastName_hw6.tar** (lastName is your last name) with all your work and submit it.

Grading Criteria

1. (10 points) Make files provided and working correctly.
2. (35 points) Program gives correct results and is robust.
3. (25 points) All implementation requirements were satisfied.
4. (20 points) The driver program works correctly (with other test files.)
5. (10 points) Analysis and Results clear and concise.