University of Massachusetts Dartmouth
Department of Electrical and Computer Engineering

ECE 160 Name: `complex.cpp, complex.h`
Project 8 due: see `http://ece160.org`

Background

The submittals for this project will be somewhat different than for previous projects or labs. Often, a programmer's task will be to write only a small part of a much larger project. For this project you will be required to create three files within one project. The files are:

- `complex.h` will contain prototypes of `complex2()`, `complex3()`, and `complexP()`
- `complex.cpp` will contain the actual code implementation for the above functions
- `testcmlx.cpp` the `main()` program with code needed to test the functions.

Although you will create three files to complete this lab, <u>you will only turn in two of the files. You are not to turn in any file which contains the main() program. Turning in the main() program will result in a grade of 0 without the possibility of resubmit.</u> The `complex.h` file should consist only of three prototype descriptions; these are the prototypes of the functions, and are detailed below. The `complex.cpp` file should consist of the functions themselves.

Overview

This project requires you to write functions to implement mathematical functions for complex numbers. A complex number consists of a real part and an imaginary part, typically described as a+bi where i is the square root of -1. Many mathematical operations may be performed on complex numbers, including addition, subtraction, multiplication and division. The first three operations (add, sub, mult) are important to understand for this programming project. If you need a review of how to work with complex numbers, you may consult:

http://www.purplemath.com/modules/complex.htm

Specifics

Start this project as you would any other project up to the point where you "Add files to project". This project will require three files (you will submit only two however). The project should be created the same way as you have in the past. Adding files now changes a bit. Up to now, you have always named the project and the .cpp file the same. This is generally ok when using only one .cpp file. With multiple files a new approach is needed. While the files may be added in any order, because of Microsoft's intellesense, it is best to do the `complex.h` first, then the `complex.cpp`, and lastly the `testcplx.cpp`.

After the project has been created in the normal way you should first, create the `complex.h` file. You may enter the file exactly as in appears below. A `.h` file is typically used to describe the prototypes for functions. In this project, the .h file will contain the prototypes for the functions implemented in the `complex.cpp` file. To add this `.h` file to the project, select Project/Add New Item. Expand Visual C++ if necessary, Click on "Code", and select "header file (`.h`)". Enter `complex.h` for the file name. Then enter code similar to the following:

```
// complex.h

void complexP(double, double);       // Print complex value

void complex2(int op,                // operation
            double *, double *,      // result Z = Z op A
            double,   double );      // value A

void complex3(int op,                // operation
            double *, double *       // result C = A op B
            double,   double,        // value A
            double,   double);       // value B
```

Next, create the `complex.cpp` file. This file will contain the actual code for doing the conversions. To add this file to the project, select Project/Add New Item. Expand Visual C++ if necessary, Click on "Code", and select "C++ File (.cpp)". Enter `complex.cpp` for the file name. Then enter code similar to the following. Obviously, you need to put some "meat" in place of the comments.

```
void complexP(double RealZ, double ImagZ)
{



      // code for this function



}


void complex2(int op,                               // operation
            double* pRealZ, double* pImagZ,      // result Z = Z op A
            double   realA, double   imagA)      // value A
{



      // code to implement the complex2() function



}


void complex3(int op,                               // operation
            double* pRealC, double* pImagC,      // result C = A op B
            double   realA, double   imagA,      // value A
            double   realB, double   imagB)      // value B
{



      //    code to implement the complex3() function



}
```

Lastly, you should enter the code for `testcplx.cpp`. Select Project/Add New Item. Expand Visual C++ if necessary, Click on "Code", and select "C++ File (.cpp)". You must ame the file with your main() function `testcplx.cpp`. Then enter code similar to the following. Since you are not turning in this file, you can put whatever code in it you want. The code below is only a sample. You should put enough test cases in this file to be sure that your three routines (`complex2()`, `complex3()`, `complexP()`) work. Pay attention to the line `#include "complex.h"`. Using quotes around a file name means the header file is in your local directory. Using < > means the header file is in a system directory. The `#define CR printf("\n")` defines a symbol (`CR`) that will be replaced by `printf("\n")` whenever it is used. This is just a shorthand way of printing out a newline.

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include "complex.h"
#define CR printf("\n")
void main()
{
       double a,b,c,d, e, f, g, h, x, y;

       g=5.2; h=7.3; complexP(g,h);CR;     // prints 5.20+7.30i
       x=2.999; y=-5.0; complexP(x,y);CR;  // prints 3.00-5.00i

       a=1.2; b=3.4; c=5.6; d=7.8;
       complex3(1,&e,&f,a,b,c,d);complexP(e,f);CR;  // prints 6.80+11.20i
       a=1.2; b=3.4; c=-5.6; d=-9.0;
       complex3(1,&e,&f,a,b,c,d);complexP(e,f);CR;  // prints -4.40-5.60i
       a=2.0; b=3.0; c=5.0; d=7.0;
       complex3(2,&e,&f,a,b,c,d);complexP(e,f);CR;  // prints -3.00-4.00i
       a=2.0; b=3.0; c=5.0; d=7.0;
       complex3(3,&e,&f,a,b,c,d);complexP(e,f);CR;  // prints -11.00+29.00i
       CR;CR;
       a=1.2; b=3.4; c=5.6; d=7.8;
       complex2(1,&a,&b,c,d);complexP(a,b);CR;  // prints 6.80+11.20i
       a=1.2; b=3.4; c=-5.6; d=-9.0;
       complex2(1,&a,&b,c,d);complexP(a,b);CR;  // prints -4.40-5.60i
       a=2.0; b=3.0; c=5.0; d=7.0;
       complex2(2,&a,&b,c,d);complexP(a,b);CR; // prints -3.00-4.00i
       a=2.0; b=3.0; c=5.0; d=7.0;
       complex2(3,&a,&b,c,d);complexP(a,b);CR; // prints -11.00+29.00i

}
```

Detailed Description of functions
(for all of the code examples below, assume all variables are of type double)

The complexP() function should output the value of the given value, with an "i" after the imaginary term. There should be no spaces or \n before or after the output. Note: a specification like `%+.2lfi` is very handy for the imaginary part.

```
g=5.2; h=7.3; complexP(g,h);      // prints 5.20+7.30i
x=2.999; y=-5.0; complexP(x,y);   // prints 3.00-5.00i
```

The `complex3()` function takes an integer operator and three complex numbers (six doubles); it provides the result of the operation (1=add, 2=sub, 3=mult). Some examples:

```
a=1.2; b=3.4; c=5.6; d=7.8;
complex3(1,&e,&f,a,b,c,d);complexP(e,f)  // prints 6.80+11.20i
a=1.2; b=3.4; c=-5.6; d=-9.0;
complex3(1,&e,&f,a,b,c,d);complexP(e,f)  // prints -4.40-5.60i
a=2.0; b=3.0; c=5.0; d=7.0;
complex3(2,&e,&f,a,b,c,d);complexP(e,f)  // prints -3.00-4.00i
a=2.0; b=3.0; c=5.0; d=7.0;
complex3(3,&e,&f,a,b,c,d);complexP(e,f)  // prints -11.00+29.00i
```

The `complex3()` function takes seven (7) arguments as follows:
- operator designation - 1=add, 2=sub; 3=mult; other=print error
- address of real part of result
- address of imaginary part of result
- real part of operand 1
- imaginary part of operand 1
- real part of operand 2
- imaginary part of operand 2

The `complex2()` function takes an integer operator and two complex numbers (four doubles). The first set of values are both one of the operands and where the result will be placed. The operator has the same meaning as in complex3(). Some examples:

```
a=1.2; b=3.4; c=5.6; d=7.8;
complex2(1,&a,&b,c,d);complexP(a,b)  // prints 6.80+11.20i
a=1.2; b=3.4; c=-5.6; d=-9.0;
complex2(1,&a,&b,c,d);complexP(a,b)  // prints -4.40-5.60i
a=2.0; b=3.0; c=5.0; d=7.0;
complex2(2,&a,&b,c,d);complexP(a,b)  // prints -3.00-4.00i
a=2.0; b=3.0; c=5.0; d=7.0;
complex2(3,&a,&b,c,d);complexP(a,b)  // prints -11.00+29.00i
```

The `complex2()` function takes five (5) arguments as follows:
- operator designation - 1=add, 2=sub; 3=mult; other=print error
- address of real part of result and first operand
- address of imaginary part of result and first operand
- real part of operand 2
- imaginary part of operand 2