Complex Functions

This is a **pair** exercise and must be competed in your **tutorial** or **lab** with your partner.

You should make sure you have completed Complex Absolute before completing this task.

In this activity you need to complete the functions `complexAdd`, `complexMultiply`, and `complexSquare`. in the provided program.

Download `complexFunctions.c`, or copy it into your current directory on a CSE system by running

```
$ cp /web/cs1511/17s2/week06/files/complexFunctions.c .
```

In this exercise you have been provided with a struct to represent a Complex Number, called `complex`. Read the provided file and understand the given data structure.

You have also been given a `main` function and three incomplete functions:

```
complex complexAdd(complex a, complex b);
complex complexMutiply(complex a, complex b);
complex complexSquare(complex c);
```

`complexAdd` takes in two complex numbers and must return a complex number that is the sum of these two numbers. Adding two complex numbers is achieved by adding the real and imaginary parts independently, like so:

$$(a + bi) + (c + di) = (a + c) + (b + d)i$$

Similarly, `complexMultiply` takes in two complex numbers and must return a complex number that is the product of these two numbers. The formula to multiple two complex numbers is as follows:

$$(a + bi)(c + di) = (ac - bd) + (bc + ad)i$$

Finally, `complexSquare` takes in one complex number and must return its square.

We've `#include` d a new header file, `<math.h>`. This declares a function, `double sqrt(double x)`, which returns the square root of `x`.

An Example

```
Enter the real part of the first number: 1
Enter the imaginary part of the first number: 17
Enter the real part of the second number: 2
Enter the imaginary part of the second number: 3
The sum is 3.00 + 20.00i.
The product is -49.00 + 37.00i.
The square of the first number is -288.00 + 34.00i.
```

To run some simple automated tests:

```
$ 1511 autotest complexFunctions
```

To run Styl-o-matic:

```
$ 1511 stylomatic complexFunctions.c
Looks good!
```

You'll get advice if you need to make changes to your code.

Submit your work with the *give* command, like so:

```
$ give cs1511 wk06_complexFunctions
```

Or, if you are working from home, upload the relevant file(s) to the `wk06_complexFunctions` activity on Give Online.