CS2211a Assignment 5

Issued on: Thursday, November 26, 2015

Due by: Monday December 7, 2015 at 11:55 pm,

Flowchart is not required in this assignment

- For this assignment, ***only electronic submission*** at owl.uwo.ca is required.
- ONLY user **Courier New** (*size = 11 pts.*)
- ***Start each question in a NEW PAGE***
- ***Write the question number is a separate line followed by an empty line***
- After finishing the assignment, you have to do the following:
  - ❖ Type your report and convert it to the PDF format (*no handwriting*),
  - ❖ The report should include:
    - o Answers to *all* questions/requirements in the assignment
    - o Enough test cases (as well as sample outputs) to demonstrate and cover all possible options in your program
    - o A copy of *all programs* that you have written
  - ❖ Prepare a soft-copy submission, including:
    - o A copy of your *typed* report
    - o All programs that you wrote (*each program in a file*)—*4 in total* (`operation_functions.c`, `operation_functions.h`, `operation.c`, and `makefile`).
      These files **MUST BE** text ASCII files. Do not submit them as PDF.
    - o ***A readme file to say how to compile and test your program.***
  - ❖ Upload the soft-copy submission file-by-file (*6 in total*), or as an archived directory.

Failure to follow the above format may cost you 10% of the total assignment mark.

- Late assignments are strongly discouraged
  - o 10% will be deducted from a late assignment (up to 24 hours after the due date/time)
  - o After 24 hours from the due date/time, late assignments will receive a zero grade.

A *complex number* is a number that can be expressed in the form $a + i\,b$, where a and b are real numbers and i is the imaginary unit, which satisfies the equation $i^2 = -1$.
In $a + i\,b$, a is called the *real part* and b is called the *imaginary part* of the *complex number*.

Complex numbers extend the concept of the one-dimensional number line to the two-dimensional complex plane by using the horizontal axis for the real part and the vertical axis for the imaginary part.

a) **(2 marks)** Declare a tag named `complex_tag` for a structure with two members, `real` and `imaginary`, of type `double`.

b) **(2 marks)** Define a type (using `typedef`) named `Complex_type` for a structure with two members, `real` and `imaginary`, of type `double`.

c) **(10 marks)** Write a function that accepts two parameters of type `complex_tag` and returns a `Complex_type` of their multiplication.

   Note that: if c_1 and c_2 are two complex numbers, where $c_1 = a_1 + i\,b_1$ and $c_2 = a_2 + i\,b_2$, then
   $c_1 \times c_2 = (a_1 \times a_2 - b_1 \times b_2) + i\,(a_2 \times b_1 + a_1 \times b_2)$

d) **(15 marks)** Write a function that accepts three parameters of type **pointer** to `complex_tag` and returns an ***integer***. The function will set the content of the third `complex_tag` pointer to be the division result of the content of the first two `complex_tag` **pointers**.

   Note that: if c_1 and c_2 are two complex numbers, where $c_1 = a_1 + i\,b_1$ and $c_2 = a_2 + i\,b_2$, then
   $c_1 \div c_2 = (a_1 \times a_2 + b_1 \times b_2) \div (a_2^2 + b_2^2) + i\,(a_2 \times b_1 - a_1 \times b_2) \div (a_2^2 + b_2^2)$, ***where*** $\boldsymbol{(a_2^2 + b_2^2) \neq 0}$
   Note that, if $\boldsymbol{(a_2^2 + b_2^2) = 0}$, return `-2`, otherwise return `0`.

e) **(20 marks)** Write a function that accepts four parameters: two parameters of type `complex_tag` and the other two of type **pointer to a pointer** to `complex_tag`. The function returns an ***integer***. The function ***allocates*** memory for two `complex_tag` variables using the `malloc` function. Make sure that the *memory allocation operation* is successful before using its output Otherwise, you need to print an error message and return `-1`. The value of one of these two allocated `complex_tag` variables will be the sum of the first two arguments, while the other will be the difference between them. A pointer to one of these two created structures will be returned to the caller via the third parameter, while the pointer to the other created structure will be returned to the caller via the fourth parameter. Upon successfully calculating the sum and difference, return `0`.

   Note that: if c_1 and c_2 are two complex numbers, where $c_1 = a_1 + i\,b_1$ and $c_2 = a_2 + i\,b_2$, then
   $c_1 + c_2 = (a_1 + a_2) + i\,(b_1 + b_2)$
   $c_1 - c_2 = (a_1 - a_2) + i\,(b_1 - b_2)$

f) **(4 marks)** Put the three functions in a file called `operation_functions.c` and their prototypes in a file called `operation_functions.h`.

g) Write a main program (`operation.c`) that
   - **(10 marks)** Declares two variables of type `complex_tag`. The value of these two variables will be initialized using the command-line arguments as 4 separate values, two for each variable.
   - **(15 marks)** Passes these initialized two structure variables (or a pointer to them, whenever appropriate) and whatever other arguments as needed to the three functions described above. Store the results in appropriate variables that you will also need to declare in the main function.
   - **(5 marks)** Prints the entered complex numbers, as well as the results of multiplication, division, addition, and subtraction operations in an easy-to-ready format.

h) **(8 marks)** Write an appropriate ***makefile*** that ***compiles*** and ***tests*** your program.

i) **(2 marks)** Provide a readme file that will tell how to use the make file to compile and/or test your program.

j) **(7 marks)** You should include as many test cases as possible to demonstrate various cases, e.g., dealing with

- ✓ two complex numbers,
- ✓ two real numbers,
- ✓ two imaginary numbers,
- ✓ a real number and an imaginary number,
- ✓ an imaginary number and a real number,
- ✓ a zero and a complex number, and
- ✓ a complex number and a zero.