Week 4 - Exercises

1. Write a program that defines a class called **ComplNum** for complex numbers with two private attributes are **real** and **imag** (double).

   a) Provide two overloaded versions for the constructor: a default constructor, and another one with two arguments to initialize attributes (require to use **`this`** keyword).

   b) Overload - and -- operators to use for **ComplNum** objects as below:

   - `object - object`
   - `object - double`
   - `double - object`
   - `object--;`
   - `--object;`

   Test all cases above in main() functions.

   c) Write a non-member friend function which accepts an **ComplNum** object and return its absolute value. Test it in main() function.

2. Write a program that ask the user to input two strings **str1, str2** which may contain whitespace characters from the console.

   a) Reserve str1 and print it out.

   For example, "Good Morning" → "gninroM dooG"

   b) Convert all characters of str2 from lowercase to uppercase and vice versa, and print it out.

   For example, "Hello World" → "hELLO wORLD"

   Do the above tasks with character array (C-type string) and C++ string class, then convert to the other.

EXTRA

3. Review of previous week's Exercise

Define a class namely **Cat** with public attributes are **name (string)**, **age (int)**.

Define another class namely **Person** with public attributes are **cat** (**Cat** class) and **name** (string).

Each class must have their own constructors.

a. Make an array of three persons, and initialize values for them.

b. Print out the information of the person who has the cat with the highest age (also print out information of that cat).

4. Review of Early Feedback Exercise

Define a class namely **BankAcc** with two public attributes are **name** (string) and **amount** (float).

a) Write a class constructor to initialize those attributes. Test it in main by creating **three** bank account objects and use the constructor to initialize data for them. Then print out all information.

b) Write a public method for the class namely **withdraw(float num)** which allow a user to withdraw money from his/her account:

- *If current amount >= num: subtract amount value by num*
- *If current amount < num: return an error code, and print out an error message.*

Test the method inside the main function.

c) Modify the program above so that it can save all data (one line for each account) into a file, whose filename is given from the command line argument (at the end of the program). If the user does not give the command line argument, don't save data.

Example Run: **./a.exe data_file.dat**