同濟大學
TONGJI UNIVERSITY

School of Software Engineering

Object-Oriented Programming OOP (2016)
Exercise 1

Please submit the source files, program results after running the .exe file and related documents. The submitted package/files should be named by “yourname_studentid_exerciseno”. Submit the package/files to your own folder on server (\\10.60.41.1). It should be the C++ folder.

Due day: 12:00 noon, Wednesday, Mar. 23, 2016

Problem 1

Create a program that counts the occurrence of the word “**that**” in a file (use the operator ‘= =’ in the string class to find the word). A sample file is given.

Problem 2

Create a *vector<float>* and put 25 numbers into it. Then square each number and put the result back into the same location in the vector. Display the vector before and after the multiplications.

Problem 3

Create two functions, one that takes a *string** and one that takes a *string&* as a parameter. Each of these functions should modify the outside string object in its own unique way. In the *main*() function, create and initialize a string object, print it, then pass it to each of the two functions, printing the results.

Problem 4

Create a *struct* that holds two string objects and one int. Use a *typedef* for the *struct* name. Create an instance of the *struct*, initialize all three values in your instance, and print them out. Take the address of your instance and assign it to a pointer to your *struct* type. Change the three values in your instance and print them out, all using the pointer.

Problem 5

We display two numbers representing a numerical sequence and then request our user to identify the next value in the sequence. For example,

The value 2, 3 form two consecutive elements of a numerical sequence.
What is the next value?

Take an example of Fibonacci sequence, 1, 1, 2, 3, 5, 8, 13 and so on. Each subsequent element is the sum of its two preceding elements.
If the user gets the correct value 5, we congratulate the user and ask whether he/she would like to try another round. For any other incorrect answer, we ask the user whether he/she would like to guess again. To make the game interesting, we keep a running score based on the number of correct answers divided by the number of guesses.

Based on the description of the game above, please present your design on how to implement the game. The key point is the design, so any formats (flowcharts, pseudo code etc.) of representing the design are accepted.
Please work in a group. When you submit your work, write down the list of the team members.