EEE102C++ Programming and Software Engineering II

Assessment4

Assessment Number	4
Contribution to Overall Marks	15%
Submission Deadline	April 29, 23:55

How the work should be submitted?

SOFT COPY ONLY !

(MUST be submitted through ICE so that the TAs can run your programs during marking.)
Make sure your name and ID are printed on the cover page of your report.

Assessment Overview

This assessment aims at testing some basic concepts of C++ programming and initiates the routine of code development using the software development process (**SDP**), namely the five main steps of the software development process:

1. Problem statement: formulate the problem.
2. Analysis: determine the inputs, outputs, variables, etc
3. Design: define the list of steps (the algorithm) needed to solve the problem.
4. Implementation: the C code has to be submitted as a separate file. Just indicate here the name of the file.
5. Testing: explain how you have tested and verified your C program.

You will need to apply this methodology to each one of the following simple exercises.

What should be submitted?

A short ***report*** (up to a few pages of texts plus C++ source codes) detailing for all the questions of the assignment. The answer for each question should follow the SDP method:

a) SDP steps 1 to 3. (30% of the total marks for that question)
b) SDP step 4 (implementation): your C++ source code including the comments. (50%)
c) SDP step 5 (testing): you will explain how you have tested the correctness of your C++ program and will include some sample runs of your C++ Programs. (20%). **Testing result must be shown by screenshot.**

The report in Microsoft Word format **(.DOCX file)**and **C++ source code (with comments)**, for all questions should also be zipped into ***a single file***. (It is a good practice to include comments in your code stating the aim of the program, what are the inputs, what are the outputs, which algorithm is used, who is the author and so on.)

EXERCISE 1 (7.5 POINTS OUT OF 15)

Derive a sub-class iFraction from the base class Fraction, which was designed by you in Exercise 2 of Assessment 2. The iFraction class represents *the mixed fractions* like

$$1\frac{2}{3}$$

Part 1: Design the sub-class **iFraction**:

1. It should have a constructor for initialisation;
2. It should contain a method to display the mixed fraction on screen;

Part 2: Design an external function **convertF** (not function member of class Fraction and iFraction) to convert the mixed fractions to improper fractions.
Hint: convertF can be the friend function of those two classes.

EXERCISE 2 (7.5 POINTS OUT OF 15)

Take the code provided for the container class (**container.h**), player class (**player.h**) and the swordsman class (**swordsman.h**), and the main function (**main.cpp**).

Part 1:

1. Read the source codes, understand the meaning and fill the blank (shown as ?????????) to make it work;
2. Generate the CRC cards of the classes: clarify the responsibilities of each class, illustrate the collaboration with others and label all the members' as public, protected and private. Then draw the hierarchy chart of them;

Part 2:

3. Design another two classes, archer and mage. Generate their CRC card and put them in the hierarchy chart drew above. Write the code for these two classes and make them work (cooperate with main function);
4. Modify main function, to have enemies with all three professions (could be randomly chosen).

Part 3 (Optional):

5. Can you involve another attribute – luck, in this game?

Notes: The factors are imperfect, make the fight too hard or too easy. Change whatever you want, but leave hints for reviewers to know what you have done.