CS 247 Software Engineering Principles — Spring 2021

Final Assessment

Due date: 15. August 2021 @ 23:59

INDIVIDUAL WORK ONLY! NO TEAM COLLABORATION!
Please expect strict plagiarism checks.

The final assessment consists of two components: a final design document for Project 3 (60 % of the grade) and assessment questions (40% of the grade).

Submission Requirements

- Final Assessment Document Part 1 — Final Design Document for Project 3 (1 PDF file)
- Final Assessment Document Part 2 — Assessment Questions (1 PDF)
  You MUST submit the C++ code solutions for Question 2 in the same assessment document.
  DO NOT create separate .cpp files!
- Project 3 Source Code (1 ZIP, from your P3 team submission)
- Project 2 Report (1 PDF, from your P3 team submission)

1 Part 1: Final Design Document for Project 3 (60%)

Submit an updated design document that outlines the final, actual design of your project, and how it differed from your design from the Project 2 report (if it did). You should include an updated UML diagram, reflecting the actual structure of your project. Do this even if it is the same as the original UML.

- Your document should provide an overview of all aspects of your project, including how, at a high level, they were implemented. If you made use of design patterns, clearly indicate where.
- Your system should employ good object-oriented design techniques, as presented in class. Include all answers to the Design Questions in the project instructions, and indicate how they differ from the answers you gave in Project 2, if they did.
- TAs are not expected to read through all of your code. Therefore, your design document should be standalone—the TA should not need to have your code opened in order to understand your document. However, if you wish to highlight certain aspects of your design, you should indicate clearly where they can be found in your code, if the TA wants to have a look.
- **THE MOST IMPORTANT ASPECT** of your design document is a discussion of how your chosen design accommodates change, as described earlier. You should also discuss the cohesion and coupling of your chosen program modules.

1.1 Length and Format of Final Design Document

We expect that your final document will be at most 5 pages long, where each page has a normal word density. Do not waste space by using larger fonts or margins and do not write wordy filler text

just to reach the page limit.

Organize your document into sections, using the following structure.

- **YOU MUST** include the list of all team members and their WatIAM ID and student numbers.
- Overview (describe the overall structure of your project, 0.5 page) [6 marks]
- Updated UML diagram (0.5 page minimum) [12 marks]
- Design (0.5 page minimum) [15 marks]
  Using POINT FORM, describe the specific techniques, STL features, and design patterns you used to solve the various design challenges.
- Resilience to Change (0.5 page minimum) [15 marks]
  Describe ONE of the key design patterns that supports the possibility of various changes to the program specification, and clearly describe how it helps with design decoupling.
- Answers to Design Questions (Revised answers from P2 Report, 1 page maximum) [12 marks]
  - How could you design your system to allow for some generated blocks to disappear from the screen if not cleared before 10 more blocks have fallen? Could the generation of such blocks be easily confined to more advanced level?
  - How could you design your program to accommodate the possibility of introducing additional levels into the system, with minimum recompilation?

Your final assessment report will be assessed according to how you solved the problem, as described in your document, and in your Project 2 UML. Your documentation will be graded according to how well you described your design (i.e., is it clear and well-communicated?).

2 Part 2: Assessment Questions (40%)

2.1 UML Diagrams [14 marks]

A car dealership is interested in creating an in-house app for customer purchases. The app permits three modes of payments: full payment, leasing, and financing. The app uses the credit scores of customers to classify them into good credit customers and bad credit customers. Good credit customers are allowed to use any of the modes of payments with zero down payment. Bad credit customers on the other hand can only do full payments or they must pay a down payment inversely proportional to their credit score. A customer is classified as a good credit customer if the credit score is above some level X. After the down payment, the remaining cost is distributed evenly according to the number of years on the contract the customer signs. The dealership would also like the app to record employee sales bonuses, which are 2% of the cost of the car sold by non-managers and 3% for managers.

Create a UML class diagram showing appropriate classes and associations for an implementation of this system.

You should only need a single UML class diagram, so half a page to a page in the document.

2.2 STL Functors and Algorithms [14 marks]

Write a small C++ program that uses an STL algorithm to read a sequence of integers from standard input and stores the integers in an STL container of your choice. The input may have repeating adjacent integers.

In your program you should do the following:

- Find all integers that are repeated adjacent to each other in the sequence.
- Sort the integers from the above step.
- Finally, output the sorted list of integers, one integer per line.

You must do all of these steps using STL functors and algorithms. Your program must include all necessary libraries and must declare and define any function or class that it uses.

Sample:

Input: 10 20 20 30 40 50 80 80 70 60 60
Found adjacent integers: 20 80 60
Sorted: 20 60 80
Output: 20
60
80

Include minimal comments in your code to highlight key aspects. Remember to simply include your C++ code in the PDF. This should take half a page to a page in the document.

2.3 Short Questions [12 marks]

Answer each of the following questions with a few sentences, and maybe a simple picture or example code fragment, whenever appropriate.

- Describe what problem the Strategy design pattern is intended to solve and provide one concrete example usage. [3 marks]
- What are some potential issues with not having proper information hiding? [3 marks]
- Describe two advantages of using the Decorator design pattern over subclassing. What is one disadvantage? [3 marks]
- Why/when are switch statements considered a “bad smell”? Describe one concrete way you can avoid them. [3 marks]