v2023.00

THE UNIVERSITY OF SYDNEY

MTRX3760 - Lab 4 Refactoring and Design

This assignment contributes 5% towards your final mark. It is to be completed individually, not in groups. Total Marks: 100.

This assignment is due before the start of your allocated lab session in one week, i.e. before the start of the Week 6 lab session for Thurs/Fri labs, and before the start of the Week 7 lab session for Monday labs. Late assignments will be subjected to the University's late submission policy unless accompanied by a valid special consideration form. Plagiarism will be dealt with in accordance with the University of Sydney plagiarism policy. **Lab submissions will not be accepted more than a week after the due date.**

Assignments will be assessed based on the following components. Incomplete submissions will have severely reduced marks:

- **Report**: Submit a .pdf report using the class Canvas site. The report can be very simple, and needs only directly address the questions laid out in the assignment.

  The cover page for your report should include your SID and tutorial section, but do not include your name to comply with the University's anonymous marking policy.

  Code appendix: The appendix of your report **must** contain a **printout** (in text format) of all source code written for the assignment. File header comments and proper formatting will be critical to making this section readable. Unintelligible code attachments will result in loss of marks.

- **Code**: Submit your code (code only, no binaries) in a .zip file via the canvas site.

Please see Lab 1 for hints on preparing your report, including how to format code for inclusion in the appendix.

Logic Simulator Redesign

This lab has the following goals:

1. To allow you to revisit a familiar design problem with new learnings covered over the first four weeks of the unit,
2. To simulate and build your understanding of a system of moderate complexity, and
3. To exercise design thinking and decision-making when faced with the many open design choices typical of a real-world engineering problem.

In the circuit simulator from Lab 2 we used hard-coded circuit designs. For this lab you will redesign the circuit simulator to load the description of the circuit from a circuit description file. This means your program should build the circuit dynamically at runtime rather than in a hard-coded manner.

You may design the circuit description language yourself, though an example of a suitable format and a means of reading it is provided in the code accompanying this lab handout.

As part of the refactor, we will lift a few simplifications applied in previous labs. We have previously allowed global constants and single-file submissions. These are not applicable to this lab:

1. Employ the “encapsulate everything” strategy of object-oriented design taught in lecture, such that there are no global constants or floating functions or data,
2. Build your solution as a multi-file project, following the guiding principle that each class should generally have its own header and implementation files, and
3. In general there should be very few constants or hard-coded circuitry, as the goal is to move those aspects of the program into the input files.

To make this lab more tractable the following new simplifications may be applied:

1. Support for subcircuits is not required,
2. You may limit the logic gates supported to AND, OR, XOR and NOT gates.

You may choose to start from your own Lab 2 solution or the one provided, or you may choose to start from scratch. Regardless, your submission should demonstrate the principles of object-oriented design and C++ coding best practices taught in lecture.

THE UNIVERSITY OF SYDNEY

[20 Marks] **Functionality:** In your report show the content of your full adder and half adder input files, and show the output of your program demonstrating that both work correctly. *Note: copy/paste the text, screenshots are unacceptable.* The program output and the code in your attached .zip file will be assessed for **correct functionality**.

[40 Marks] **Code Quality and Development Process**: Include in your report your git commit history as reported by:

```
git log --oneline
```

Note: copy/paste the text, screenshots are unacceptable

Your code and commit history will be assessed based on **style** and **coding and development best practices** as taught in lecture.

[40 Marks] **Design**: In your report provide a UML class diagram for your design. You need not include member variables and functions. This diagram and your code will be assessed based on appropriate use of the **design principles** taught in lecture.

Self-Assessment

[+5 Bonus] In your report estimate the level of achievement of your submission. Show estimates for all component grades as well as the total. If your estimated total is within 5 marks of the correct score you will be awarded 5 bonus marks. Ignore late penalties when estimating.

/20 Functionality
/40 Code quality
/40 Design
/100 Total