# UNIVERSITY OF LONDON

# BSc EXAMINATION 2024

For Internal Students of
Royal Holloway

## DO NOT TURN OVER UNTIL TOLD TO BEGIN

## CS2800: Software Engineering
## CS2800R: Software Engineering – for FIRSTSIT/RESIT CANDIDATES

Time Allowed: **TWO hours**

Please answer **ALL** questions

Calculators are not permitted

1. For each of the following pairs of related software engineering concepts you must:

   - Briefly *describe* each of the two concepts and why they are important in software engineering. This should be enough to introduce the concept to a new student on CS2800.

     The two descriptions could have points in common, which might then also be a part of their connection.

     *Each description could be about six lines of text.*

   - Show that you understand how the two concepts are *connected*.

     For example, they have the same or contrasting goals, or they may be techniques that rely on each other to work. This needs careful thought.

     *A good answer could be about four lines of text.*

   (a) *Literate Programming* and *Programming Standards*. [12 marks]

   (b) *Aggregation Relationship* and *Composition Relationship*. [8 marks]

   (c) *Good Code* and *Test Driven Development (TDD)*. [12 marks]

   (d) *Code Release* and *Incremental Business Value*. [8 marks]

2. Consider the following code sample:

```
1   private float money;
2   private float interest;
3   public int Foo(float value) throws NeverException {
4    int years = -1;
5    if (interest <= 0f || money <= 0f) {
6     throw new NeverException();
7    }
8    if (value > 0) {
9     for (years = 0; money < value; years = years + 1)
10     value = value * (1.0f + interest / 100.0f) ;
11   } else {
12    years = 0;
13   }
14   return years;
15  }
16
17  /*
18   private float bar (v , i){
19    return v * (1.0 f + i / 100.0 f);
20   }
21  */
22
```

(a) Draw the control flow graph for the method `Foo` from the code above. [8 marks]

(b) Calculate the cyclomatic complexity of the method `Foo` from the code above. [3 marks]

(c) The method `Foo` in the code above is unit tested by running it with different values set for `money`, `value` and `interest`. Each unit test will execute some of the statements of the method.
Give a minimum number of tests (as values for `money`, `value` and `interest`) that (together) execute all the statements of the method `Foo`.
To prove coverage, list the line numbers of the statements that are executed by each of your tests. [4 marks]

(d) Why should we worry about the too few comments smell? What might it tell us about the software engineer who wrote the code? [4 marks]

(e) Why should we worry about the dead code smell? What might it tell us about the software engineer who wrote the code? [4 marks]

3. (a) What is the effect of `git commit` in the Git Revision Control System? [3 marks]

   (b) What is a *tag* in the Git Revision Control System? [3 marks]

   (c) How does a *tag* differ from a *branch* in the Git Revision Control System? [4 marks]

   (d) In the the Git Revision Control System what is the effect of the command `git revert <commit-SHA>`? [4 marks]

   (e) What is the effect of the command `git push`? Give TWO reasons why it is not a good idea to immediately follow each `git commit` by a `git push` . [4 marks]

   (f) Explain why we should not see any of the following Git log messages.

   - Fixed the Checkstyle Violations and added the missing Javadoc. [2 marks]
   - Successful Commit. The fault is now fixed. [2 marks]
   - Saving and pushing code because it is lunchtime. Will complete the test case later this afternoon. [2 marks]

4. This question is about improving the design of the following Puzzle Game.

A Puzzle game: A nine by nine grid of cells. Some rules about filling them in. There are three kinds of rules.

- *Region*: A `List` of grid locations which must all contain different digits.
- *Killer*: A *Region* rule **together with** a total for all of the cells in the *Region*.
- *Given*: A single grid location that is pre-filled with a value.

The grid and the rules are displayed to the user, who enters digits into cells.

---

A `Game` has a `Set` of `Rules` and a `Position`. A `Position` contains an array of `GridCell` objects, and a count of filled cells.

Each `GridCell` has a current value that is zero if the cell is not filled. Each `GridCell` object has a list of the `Rules` it is subject to (part of). Every `GridCell` object is subject to at least one rule.

If a `GridCell` is subject to a *Given* rule then its value is fixed, and the user cannot change it.

---

SIX main classes: Game, Position, GridCell, RegionRule, KillerRule, GivenRule.

(a) Draw a UML class diagram for this model, adding any appropriate interfaces and base classes. Include appropriate multiplicities. Associations may be from a class to itself.

*Include relevant attributes and responsibilities for each class. Generic responsibilities like getters, setters, and toString should be omitted.* [4 marks]

(b) Why would we modify the design to use a Rule Factory? [3 marks]

(c) Why should we modify the design by using a Digits class? [3 marks]

(d) How should State be used in this project for Grid objects? [3 marks]

**END**