SFWRENG 3S03 Software Testing (2025 Winter)
Assignment 2

Administrative information

Weight of the assignment: this assignment is worth 10% of your final grade. It consists of three questions for a total of 70 marks, and a quality multiplier (1.0-0.5) that reflects the overall quality of your report.

How to complete the assignment?

- Answer all questions.
- Submit your solutions on Avenue.
- Submit your report with your answers as a single PDF document; and a zip file containing your code and tests for Questions 1 and 2.
- In your PDF document, report the key points and the key artifacts (ChatGPT prompts, etc). Argue your points when asked to.

Groupwork

You can work in a group of up to three people, although you don't have to. You can also switch groups and work in different groups than in the first assignment. If you work in a group: everyone in the group needs to submit the very same PDF file (due to administrative reasons).

Due date: March 7, 2025, at 11:59 PM ET.

1. Consider the following class `RomanNumerals`, which implements basic functionality for processing roman numerals.

Hand in your unit tests in a zip file. Annotate your unit tests with comments to explain what the tests should be doing. **In your report**, explain your decisions, justify your choices, and provide evidence when needed (e.g., in question *b*).

[27.5 marks]

(a) [*2 marks*] The code contains a small bug. What is the bug? How did you find it? Fix the bug, and then use the repaired code for the remaining parts of this assignment.

(b) [*15.5 marks*] Write a comprehensive set of unit tests for this class. Implement the unit tests using JUnit (or a similar tool). Hand in your unit tests, as well as a summary in your report (e.g., a screenshot) that shows the results of executing these tests.
Marking scheme: 15.5 marks for the unit tests.

(c) [*10 marks*] Calculate the statement coverage, branch coverage, condition coverage and MC/DC coverage for your unit tests. Use a coverage tool like EclEmma, or similar, for the calculations (though it is acceptable not to use a tool only for MC/DC coverage). Submit a summary/screenshot showing the results of calculations.
Note: it is not required to obtain 100% coverage, but if you do not have 100% coverage, please briefly explain why you think this is the case.

```
package roman;

public class RomanNumerals {

  public static String roman(int number) {
    int x = (number % 100) / 10;
    int i = (number % 10) / 1;
    return romanForDigit(i, 'I', 'V', 'X');
  }

  private static String romanForDigit(int digit, char one, char five, char ten) {
    if (digit == 0) return "";
    if (digit <= 3) return repeatedChar(digit, one);
    if (digit == 4) return "" + one + five;
    if (digit <= 8) return five + repeatedChar(digit-5, one);
    return "" + one + five + ten;
  }

  private static String repeatedChar(int count, char c) {
    String result = "";
    for (int i = 0; i < count; i++) result += c;
    return result;
  }
}
```

2. Below you will find the code for a Java class called `MinElement`, which implements `min()`, a method that calculates the minimum element of a generic list. Develop a set of unit tests for `MinElement`, focusing only on normal returns (i.e., ignore exceptions). **Hand in your unit tests in a zip file.** Annotate your unit tests with comments to explain what the tests should be doing. **In your report**, explain why you think this set of tests is comprehensive for the purposes of testing `MinElement`.
[27.5 marks]

Marking scheme: 20 marks for the unit tests, 7.5 marks for the explanation.

```
import java.util.*;

public class Min{
  /**
    * Returns the minimum element in a list
    * @param list Comparable list of elements to search
    * @return the minimum element in the list
    * @throws NullPointerException if list is null or
    *         if any list elements are null
    * @throws ClassCastException if list elements are not mutually comparable
    * @throws IllegalArgumentException if list is empty
    */
    public static <T extends Comparable<? super T>> T min (List<? extends T> list){
       if (list.size() == 0){
          throw new IllegalArgumentException ("Min.min");
       }

       Iterator<? extends T> itr = list.iterator();
       T result = itr.next();

       if (result == null) throw new NullPointerException ("Min.min");

       while (itr.hasNext()){   // throws NPE, CCE as needed
           T comp = itr.next();
           if (comp.compareTo (result) < 0){
               result = comp;
           }
       }
       return result;
    }
}
```

3. IEEE 829 is a standard for Software and System Test Documentation. This rather boring document gives you a standard, repeatable way to document your software and system tests. You can access the standard with a McMaster IP address from IEEE Xplore.[1]
(Note that 829 has been superseded by a more up to date standard, IEEE 29119, but that's irrelevant for the purposes of this assignment.)
[15 marks]

Note: YOU ARE NOT EXPECTED TO READ THE WHOLE STANDARD!

(a) [*3 marks*] Using your favourite generative AI technology (e.g., ChatGPT, Bard), identify three reasons to use IEEE 829 to document your tests/testing.
(b) [*3 marks*] What is good about the suggestions given in your answer to (a)? What is bad about it?
(c) [*3 marks*] Without using generative AI, come up with three *different* reasons to use IEEE 829.
(d) [*3 marks*] Using your favourite generative AI technology, identify three reasons *not* to use IEEE 829 to document your tests/testing.
(e) [*3 marks*] Without using generative AI, come up with three *different* reasons *not* to use IEEE 829.

[1] https://ieeexplore.ieee.org/document/4578383

4. Overall quality of the report

Your report should be of reasonable quality. Your intent should be to report your experiences and argue your points, not to copy-paste template text.
To obtain your final mark, the sum mark of Tasks 1-2-3 will be scaled as follows.

Quality	Example characteristics	Multiplier
Good	Concisely and precisely reports artifacts, results, and if the apply, assumptions and limitations. Clearly argues points. Figures and images are of proper resolution. Uses proper English.	1.0
Medium	Reports artifacts and results at an acceptable level. Makes points without solid arguments. Awkward and hard-to-follow figures and images. Typos in text.	0.75
Poor	Clearly lacks effort at producing a quality report. Only reports data; interpretations and arguments are completely missing. Figures and images are of poor resolution. Poor English.	0.5