Written Examination
DIT636 / DAT560
Software Quality and Testing
August 27, 2025; 14:00 – 18:00

Course Responsible/Examiner: Gregory Gay
E-Mail: ggay@chalmers.se
Phone: +46 73 856 77 93
Examination Hall Visits: 15:00, 16:30
Allowed aids: No notes or other aids are allowed.
Grading Scale: 0-49 (U), 50-69 (3), 70 - 85 (4), 86-100 (5)
Examination Review: On request

There are a total of 9 questions and 100 points available on the test. On all essay type questions, you will receive points based on the quality of the answer - not the quantity. Illegible answers will not be graded.

Question 1 (Warm Up) - 10 Points

Multiple solutions may apply. Select all that are applicable. True/False worth 1 point each, multiple choice worth 2 points each.

1. For the expression `(!a && !c) || (b || c)`, the test suite **(a, b, c) = {(T, F, T), (F, T, T), (T, T, T), (F, F, F)}** provides:
    a. MC/DC Coverage
    b. Decision Coverage
    c. Basic Condition Coverage
    d. Compound Condition Coverage

2. MC/DC coverage subsumes branch coverage.
    a. True
    b. False

3. An equivalent mutant always returns the same output as the original program.
    a. True
    b. False

4. A typical distribution of test types is 50% unit tests, 40% system tests, and 10% GUI/exploratory tests.
    a. True
    b. False

5. You have designed the software for an ATM to meet the following requirement: “Before money is dispensed, the transaction amount shall be compared to the amount of money in the machine. If the transaction total is greater than or equal to the amount of remaining money, the transaction will be canceled. No funds shall be dispensed, and no changes shall be made to a user's account.” Which type of property is this?
    a. Correctness
    b. Safety
    c. Reliability

6. Validation is the process of ensuring that an implementation matches its specification.
    a. True
    b. False

7. You are designing a service that regulates the temperature of a fuel tank. Which of the following quality attributes are the most important?
    a. Availability
    b. Performance
    c. Scalability

    Briefly (1-2 sentences), explain why this is the most important.

Question 2 (Quality Scenarios) - 10 Points

Consider a video streaming service (e.g., Netflix, Disney Plus). This service:

- Offers a library of videos that can be watched, including both TV series and movies. This library can change over time (e.g., titles can be added or removed, or metadata can be changed).
- Users can select a video and watch it.
- Videos can support different audio and subtitle options.
- Users can maintain a list of videos that they want to save for later.
- The service maintains information on whether a video has been fully watched or not, indexing the timestamp where a user left off if they did not complete a video. If a user selects an unfinished video, it should resume at that timestamp.
- Because this is a subscription service, all users must have accounts. The account details can be managed, including the user's password, address, e-mail address, and credit card number.

This service must provide functionality in a high quality manner to thousands of concurrent visitors.

Create one **reliability** and one **availability** scenario for this system, with a Description, System State, System Environment, External Stimulus, Required System Response, and Response Measure for each.

Question 3 (Testing Concepts) - 8 Points

Choose one of the stages of testing that we covered: unit, system, or exploratory testing.

Explain in your own words what that stage is, how systems are tested in that stage, how it differs from the other listed stages of testing, and the types of faults that are most likely to be exposed by that stage that would be missed during the other stages.

Question 4 (Test Design) - 12 Points

Recall the video streaming service discussed in Question 2. One of its features is that videos can support different audio and subtitle options.

Internally, the following method is invoked each time a user changes their options:

```
public Boolean changeOptions (Session session, String audioLanguage,
integer audioSpeakers, String subtitleLanguage) throws Exception
```

A `Session` is an object with the following fields:

- `userID`: String, represents the ID of the logged-in user.
- `videoID`: String, represents the ID of the video currently being watched.
- `timestamp`: Integer, represents the point in the video (in seconds) when the change to the options was requested.

`audioLanguage` and `subtitleLanguage` represent the chosen languages for the audio and subtitles. Note that not all languages are available for all videos for both options, and that the audio and subtitle options do not have to be the same language.

`audioSpeakers` represents the number of speakers to output audio to (for surround sound).

You can assume that there are databases storing information on the user and on the video (including the available languages and speaker options).

The method should return “true” if the changes are applied successfully or “false” otherwise. The method can also throw an exception if an error occurs during execution.

Perform functional test design for this method.

1. **Identify choices (controllable aspects that can be varied when testing)**
2. **For each choice, identify representative input values.**
3. **For each value, apply constraints (IF, ERROR, SINGLE) if they make sense.**

You do not need to create test specifications or concrete test cases. For invalid input, **do not** just write “invalid” - be specific.

If you wish to make any additional assumptions about the functionality of this method, state them in your answer.

Question 5 (Exploratory Testing) - 8 Points

Exploratory testing typically is guided by “tours”. Each tour describes a different way of thinking about the system-under-test and prescribes how the tester should act when they explore the functionality of the system.

1. Describe one of the tours that we discussed in class **other than the supermodel tour**.
2. Consider the video streaming service discussed in Question 2.

   Describe three distinct sequences of interactions with one or more functions of this system you would explore during exploratory testing of this system, based on the tour you described above. Explain the interactions you would take when executing that sequence, and why those actions fulfill the goals of that tour. These sequences should be different from each other (i.e., limited overlap).

Question 6 (Unit Testing) - 9 Points

Consider the audio/subtitle options method that you developed test specifications for in Question 4:

```
public Boolean changeOptions (Session session, String audioLanguage,
integer audioSpeakers, String subtitleLanguage) throws Exception
```

Based on your test specifications, write three JUnit-format test cases:

1. Create one test case that checks a normal usage of the method, where the options are applied successfully.
2. Create one test case that checks a normal usage of the method, where the chosen options cannot be applied.
3. Create one test case reflecting an error-handling scenario (an exception is thrown).

Question 7 (Structural Testing) - 16 Points

The following function returns true if you can partition an array into one element and the rest, such that this element is equal to the product of all other elements excluding itself.

For example:

- **`canPartition([2, 8, 4, 1])`** returns true (8 = 2 * 4 * 1)
- **`canPartition([-1, -10, 1, -2, 20])`** returns false.
- **`canPartition([-1, -20, 5, -1, -2, 2])`** returns true (-20 = -1 * 5 * -1 * -2 * 2)

```
1. public static boolean canPartition(int[] arr) {
2.       Arrays.sort(arr);
3.       int product = 1;
4.       if ((Math.abs(arr[0]) >= arr[arr.length-1]) || arr[0] == 0) {
5.           for (int i = 1; i < arr.length; i++){
6.               product *= arr[i];
7.           }
8.           return arr[0] == product;
9.       } else{
10.         for (int i = 0; i < arr.length-1; i++){
11.             product *= arr[i];
12.         }
13.         return arr[arr.length-1] == product;
14.      }
15.   }
```

1. Draw the control-flow graph for this function. You may refer to line numbers instead of writing the full code.
2. Identify test input that will provide statement and branch coverage.

   You do not need to create full unit tests, just supply input for the function.

   For each input, list the line numbers of the statements covered as well as the specific branches covered (use the number and T/F, i.e., “3-T” for the true branch in line 3).

   Do not use the test input from the examples above. Come up with your own input.

Question 8 (Data Flow Testing) - 12 Points

Using the same code from Question 7:

1. Identify the def-use pairs for all variables.
2. Identify test input that achieves all def-use pairs coverage.

   You do not need to create full unit tests, just supply input for the function. Again, do not reuse the sample input (however, you can reuse your own input from the previous question).

   For each input, explain which def-use pairs are covered by that input.

Note: You may treat arrays as a single variable for purposes of defining DU pairs. This means that a definition to **`arr[0]`** or to array **`arr`** are both definitions of the same variable, and references to **`arr[0]`** or **`arr.length`** are both uses of the same variable.

Question 9 (Mutation Testing) - 15 Points

This function checks whether it is possible to insert a single character into string `s1` to transform it into string `s2`:

```
1. public boolean oneEditInsert(String s1, String s2) {
2.     int index1 = 0;
3.     int index2 = 0;
4.     while (index2 < s2.length() && index2 < s1.length()){
5.         if (s1.charAt(index1) != s2.charAt(index2)) {
6.             if (index1 != index2) {
7.                 return false;
8.             }
9.             index2++;
10.        } else {
11.            index1++;
12.            index2++;
13.        }
14.    }
15.    return true;
16. }
```

Answer the following three questions for **each** of the following mutation operators:

- Relational operator replacement (ror)
- Arithmetic operator replacement (aor) (including short-cut operators)
- Constant for constant replacement (crp)

1. Identify all lines that can be mutated using that operator.
2. Choose **one** line that can be mutated by that operator and create **one** non-equivalent mutant for that line.
3. For that mutant, identify test input that would detect the mutant. Show how the output (return value of the method) differs from that of the original program.