Written Examination
DIT636 / DAT560
Software Quality and Testing
June 11, 2025; 8:30 – 12:30

Course Responsible/Examiner: Gregory Gay
E-Mail: ggay@chalmers.se
Phone: +46 73 856 77 93
Examination Hall Visits: 10:00, 11:30
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 || b) && !(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. During exploratory testing, the bad neighborhood tour recommends running functionality with deliberately malformed input to test the system's resilience.
    a. True
    b. False

3. Validation is the process of ensuring that an implementation meets the requirements of its users.
    a. True
    b. False

4. Path Coverage is unsatisfiable without placing a bound on the number of loop executions.
    a. True
    b. False

5. You have designed the software for an object detection system in a vehicle to meet the following requirement: "Objects within 2m of the camera must be correctly identified in 99.99999% of invocations of the method." Which type of property is this?
    a. Correctness
    b. Reliability
    c. Robustness

6. An equivalent mutant is a mutant that produces the same result as the original program for the current test suite.
    a. True
    b. False

7. You are designing an AI that controls enemy actions in a role-playing game. Which one of the following quality attributes would be of most importance to you?
    a. Reliability
    b. Availability
    c. Performance

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

Question 2 (Quality Scenarios) - 10 Points

Consider a web-based job application system. This system offers the following functionality:

- Employers can post jobs that job-seekers can apply to.
    - A job must have a description, a location (a job can also be “remote”, instead of having a specific location), and an estimated salary.
    - A job can be listed with 0 or more qualifications that must be met by applicants. A qualification can be:
        - A required skill.
        - A university degree in a specific area.
        - A number of years in a particular type of job role.
    - A job listing also has an expiration date, after which, no more applications will be accepted.
    - Internally, all jobs are assigned a unique ID.
- Job-seekers have a profile, which includes:
    - A list of jobs they have applied for.
    - A list of current and past jobs, including the role, company, and duration of time in that role.
    - A list of skills, chosen from a set of pre-existing keywords.
    - A list of attained university degrees.
    - Internally, all job-seekers are assigned a unique ID.
- Job-seekers can register for an account.
    - They create an initial profile as part of the registration process.
- After registration, job-seekers can edit their profile.
- Job-seekers can apply for posted jobs.
    - When they do so, their qualifications are checked against the required qualifications for the position.
    - If they do not qualify, they can still apply, but must fill in an explanation about the missing qualifications.

Create one **reliability** and one **performance** 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 following stages: unit, system (integration), and exploratory testing.

- Explain how systems are tested at the chosen stage.
- Explain the primary differences between that stage and the other two stages.
- Explain the types of faults that are most likely to be exposed by the chosen stage.

Question 4 (Test Design) - 12 Points

Recall the job application system discussed in Question 2.

The following method is invoked when an employer creates a job:

```
public UUID createJob (String description, String location, String[]
qualifications, float salary, Date expiration) throws Exception
```

The method returns a UUID, a unique identifier, for the job as long as the job is successfully created. If the job is not created, an exception should be thrown. An exception should be thrown if:

- Any of the parameters fails validation (is invalid or malformed in some way).
- If this job already exists (if the employer already has a job with the same description, location, and expiration date).

Refer to Question 2 for information about each parameter. If any aspect of the parameters is not stated or sufficiently explained, you may make assumptions. State all assumptions in your solution.

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 job application system described 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 job creation method that you developed test specifications for in Question 4:

```
public UUID createJob (String description, String location, String[]
qualifications, float salary, Date expiration) throws Exception
```

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

1. Create one test case where the job is successfully created.
2. Create one test case where the job is rejected because it already exists.
3. Create one test case where one or more input parameters fail validation.

Question 7 (Structural Testing) - 16 Points

This function takes an array of strings. It will then replace any duplicate entries (ignoring case) with the string “DUPLICATE” and any empty or null strings with “EMPTY”.

```
1.  public String[] void removeDuplicates(String[] arr) {
2.       if (arr == null || arr.length == 0) {
3.               return arr;
4.       }
5.       for (int i = 0; i < arr.length; i++) {
6.           if (arr[i] != null && !arr[i].isEmpty()) {
7.               for (int j = i + 1; j < arr.length; j++) {
8.                    if (arr[j] != null &&
                              arr[i].toUpperCase().equals(
                              arr[j].toUpperCase())) {
9.                        arr[j] = "DUPLICATE";
10.                   }
11.               }
12.           } else {
13.               arr[i] = "EMPTY";
14.           }
15.       }
16.       return arr;
17.   }
```

For example:

- [“Bob”, “Jill”] -> [“Bob”, “Jill”]
- [“Bob”, “Jill”, “BOB”] -> [“Bob”, “Jill”, “DUPLICATE”]
- [“Bob”, “”, null] -> [“Bob”, “EMPTY”, “EMPTY”]

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 branch and basic condition coverage. You do not need to create full unit tests, just supply input for the function.

   For each input, list:
   a. all specific branches covered (use the line number and T/F, i.e., “6-T” for the true branch of line 6).
   b. all specific conditions covered, and the value of those conditions (e.g., Line 2: “arr == null”, true)

You must provide this information. Test input with no explanation will not be accepted.
Do not use the test input from the examples above. Come up with your own input.

Question 8 (Mutation Testing) - 15 Points

Consider the same code from Question 7:

```
1.  public String[] void removeDuplicates(String[] arr) {
2.       if (arr == null || arr.length == 0) {
3.               return arr;
4.       }
5.       for (int i = 0; i < arr.length; i++) {
6.           if (arr[i] != null && !arr[i].isEmpty()) {
7.               for (int j = i + 1; j < arr.length; j++) {
8.                    if (arr[j] != null &&
                              arr[i].toUpperCase().equals(
                              arr[j].toUpperCase())) {
9.                        arr[j] = "DUPLICATE";
10.                   }
11.               }
12.           } else {
13.               arr[i] = "EMPTY";
14.           }
15.       }
16.       return arr;
17.   }
```

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.

Question 9 (Finite State Verification) - 12 Points

Temporal Operators: A quick reference list. p is a Boolean predicate or atomic variable.

- G p: p holds globally at every state on the path from now until the end
- F p: p holds at some future state on the path (but not all future states)
- X p: p holds at the next state on the path
- p U q: q holds at some state on the path and p holds at every state before the first state at which q holds.
- A: for all paths reaching out from a state, used in CTL as a modifier for the above properties (AG p)
- E: for one or more paths reaching out from a state (but not all), used in CTL as a modifier for the above properties (EF p)

An LTL example:

- G ((MESSAGE_STATUS = SENT) -> F (MESSAGE_STATUS = RECEIVED))
- It is always true (G), that if the message is sent, then at some point after it is sent (F), the message will be received.
    - More simply: A sent message will always be received eventually.

A CTL example:

- EG ((WEATHER = WIND) -> AF (WEATHER = RAIN))
- There is a potential future where it is a certainty (EG) that - if there is wind - it will always be followed eventually (AF) by rain.
    - More simply: At a certain probability, wind will inevitably lead to eventual rain. (However, that probability is not 100%)

Consider a warehouse system where a robot prepares shipments for delivery to a customer. A finite state model of this system has the following state variables:
Order_Status: {Placed, Packaging, Shipped}
Robot_Status: {Idle, Working}
Robot_Arm_Status: {Idle, Moving, Grabbing, Holding, Releasing}
Robot_Leg_Status: {Idle, Moving, Standing}
Packaging_Status: {No Box, Box Requested, Box Closed, Box Open, Box Sealed}

1. Write two safety properties (something must happen or not happen in a specific way and/or specific time) for this model and formulate them in LTL or CTL. These properties must exercise different scenarios, and not just have variable names changed.
2. Write two liveness properties (something must eventually happen) for this model and formulate them in LTL or CTL. These properties must exercise different scenarios, and not just have variable names changed.