Written Examination
DIT636/DIT635/DAT560 – Software Quality and Testing
August 28, 2024; 14:00 – 18:00

Course Responsible/Examiner: Gregory Gay
E-Mail: ggay@chalmers.se
Phone: +46 73 856 77 93
Examination Hall Visits: 15:30, 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.

1. For the expression `(!a && !c) && (a || b)`, 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. In a web-based store, the fed-ex tour could help detect issues with the process of fulfilling an order (i.e., ensuring that a placed order has been shipped with no issues).
    a. True
    b. False

3. Validation ensures that an implementation meets the actual requirements of its users.
    a. True
    b. False

4. Mutation is considered useful because a sequence of small code changes can approximately model a larger fault in the program.
    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. Robustness

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

7. You are designing an online multiplayer role-playing game. Which of the following quality attributes would be of most importance to you?
    a. Availability
    b. Reliability
    c. Scalability

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

Question 2 (Quality Scenarios) - 10 Points

Consider a web-based discussion forum. This system offers the following functionality:

- Registered users can post discussion topics and reply to existing topics.
- A discussion topic can optionally include a poll (users can select one of a set of responses to a user-specified question, a summary chart is displayed showing which responses that users selected).
- Users can subscribe to selected topics. When a new reply is made, subscribed users get a notification.
    - Users are auto-subscribed to topics they create.
- Administrators can create, edit, and delete “Discussion Boards”, which are areas where discussion topics can be created.
    - For example, they might create a “Movies” board, where discussion topics about movies should be posted.
- Administrators can delete topics or lock topics to prevent replies from being posted.
- Account details can be managed, including e-mail address, displayed username, password, and notification settings.

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

- Define integration testing and exploratory testing.
- Explain how systems are tested at each state.
- Explain how the two stages differ.
- Explain the type of faults that are exposed by each of the two stages.

Question 4 (System Testing) - 12 Points

Recall the web-based discussion forum discussed in Question 2.

Administrators can create “Discussion Boards”, which are areas where discussion topics can be created. For example, they might create a “Movies” board, where discussion topics about movies should be posted.

Discussion boards have the following attributes:

- A name
- A description
- Boards can be public or private (only accessible to administrators)

Internally, the following function is invoked when a board creation is requested:

```
public boolean createBoard (String name, String description, Boolean public)
```

The function returns TRUE if the user successfully created the discussion board. It returns FALSE if not. An exception can also be thrown if there is an error.

A board will be registered under the following conditions:

- The user requesting board creation is an administrator.
- The board does not already exist.
- The board name and description do not contain any banned words.

This function connects to a user database and a JSON file containing a list of existing boards.

Perform category-partition testing for this function.

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 discussion forum system from Question 2. Describe three sequences of interactions with one or more functions of the 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.

Question 6 (Unit Testing) - 9 Points

Consider the discussion board creation function that you developed test specifications for in Question 4:

```
public boolean createBoard (String name, String description, Boolean public)
```

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

1. Create one test case that checks a normal usage of the method, with a true outcome.
2. Create one test case that checks a normal usage of the method, with a false outcome.
3. Create one test case reflecting an error-handling scenario (an exception is thrown).

Question 7 (Structural Testing) - 16 Points

This method takes two positive integers, start and end, as input. The method will find and print the sum of all prime numbers between start (inclusive) and end (inclusive).

```
1.  public int sumOfPrimesInRange (int start, int end) {
2.      long sum = 0; // Initialize the sum
3.      for (int num = start; num <= end; num++) {
4.          Boolean isPrime = true;
5.          if (num < 2) {
6.              isPrime = false;
7.          }
8.          for (int i = 2; (i * i) <= num; i++) {
9.              if (num % i == 0) {
10.                 isPrime = false;
11.             }
12.         }
13.         if (isPrime == true) {
14.             sum += num;
15.         }
16.     }
17.     return sum;
18.  }
```

For example `sumOfPrimesInRange(0, 10)` returns 17.

1. Draw the control-flow graph for this program. 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 line number and T/F, i.e., “3-T” for the true branch of line 3).

Question 8 (Data Flow Testing) - 12 Points

This method computes the longest common sequence of characters between two strings:

```
public String findLongestCommonSequence(String s1, String s2) {
   String result = "";
   for (int length = s1.length(); length > 0; length--) {
       int startIndex = 0;
       while (startIndex + length <= s1.length()) {
           String current = s1.substring(startIndex, startIndex + length);
           if (s2.contains(current)) {
               result = current;
               break;
            }
            startIndex++;
       }
       if (result.length() != 0) {
           break;
       }
   }
   return result;
 }
```

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

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 takes an array of strings as input and returns the longest common prefix among all the strings. If there is no common prefix, it returns an empty string.

```
1.  public String findLongestCommonPrefix (String[] strs) {
2.      if (strs == null || strs.length == 0) {
3.          return ""; // Empty array, no common prefix
4.      }
5.      String commonPrefix = strs[0]; // Initialize with the first string
6.      for (int i = 1; i < strs.length; i++) {
7.          String current = strs[i];
8.          int j = 0;
9.          while (j < commonPrefix.length() && j < current.length() &&
                         commonPrefix.charAt(j) == current.charAt(j)) {
10.             j++;
11.         }
12.         commonPrefix = commonPrefix.substring(0, j); // Update prefix
13.         if (commonPrefix.isEmpty()) {
14.             break; // No common prefix, exit loop
15.         }
16.     }
17.     return commonPrefix;
18.  }
```

For example:
`findLongestCommonPrefix(["flower", "flour", "flight"])` returns “fl”
`findLongestCommonPrefix(["flower", "plant", "bee"])` returns “”

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.