Written Examination
DIT636/DAT560 – Software Quality and Testing
June 5, 2024; 8:30 – 12:30

Course Responsible/Examiner: Gregory Gay
E-Mail: ggay@chalmers.se
Phone: +46 73 856 77 93
Examination Hall Visits: 9:30, 10:30
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) && (b && c)`, the test suite **(a, b, c) = {(T, F, T), (F, T, T), (T, T, T), (T, 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 would be a good choice to detect issues with the process of fulfilling a placed order (i.e., ensuring that a placed order has been shipped with no issues).
    a. True
    b. False

3. Path coverage subsumes MC/DC coverage.
    a. True
    b. False

4. Exploratory testing is a cost-effective way to detect integration faults.
    a. True
    b. False

5. In a genetic algorithm, which of the following concepts are employed as part of creating a new population?
    a. Crossover
    b. Cooldown
    c. Tournament Selection
    d. Fitness Function

6. Detecting equivalent mutants is an NP-hard problem.
    a. True
    b. False

7. You are designing a chatbot that makes suggestions regarding which stocks to purchase, based on a series of questions. No information is stored between sessions. Which one of the following quality attributes is the most important?
    a. Security
    b. Performance
    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 **performance** and one **scalability** 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 unit testing and integration 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. To post topics or replies, a user must register for an account.

Internally, the form filled out to register the account is sent to the following function:

```
public boolean registerUser (String emailAddress, String userID, String
password, Date dateOfBirth)
```

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

An account will be registered under the following conditions:

- An account must not be registered already for that e-mail address.
- The e-mail address must be correctly formatted (contains “@” and a domain with a valid top-level domain, e.g., .se or .com).
- The user ID must not contain any banned words.
- The user must be at least 13 years old.

This function connects to a user database, where records are indexed by the e-mail address.

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 guidebook 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 user registration function that you developed test specifications for in Question 4:

```
public boolean registerUser (String emailAddress, String userID, String
password, Date dateOfBirth)
```

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 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 “”

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

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. 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 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.   }
```

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.