Written Examination
DIT635 – Software Testing and Analysis
August 23, 2021; 8:30 – 12:30

Name:

Course Responsible/Examiner: Gregory Gay
E-Mail: ggay@chalmers.se
Phone: Not used
Examination Hall Visits: Not applicable
Allowed aids and additional information: You may refer to the textbook, slides, and other course materials. However, do not quote directly from these materials (answer in your own words). You may not collaborate with others.

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. You may answer either in this document or in a separate document.
Grading Scale: 0-49 (U), 50-85 (G), 86-100 (VG)
Examination Review: Online (via e-mail)

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, F, T), (F, T, T), (T, F, F)}** provides:
    a. MC/DC Coverage
    b. Decision Coverage
    c. Basic Condition Coverage
    d. Compound Condition Coverage

2. Verifying that a system meets its specification is sufficient to determine if the users' needs have been met.
    a. True
    b. False

3. All DU paths coverage requires that all paths between each definition and each of its usages be covered by at least one test case.
    a. True
    b. False

4. A banking website that displays an error message and stops all normal operations when a database connection cannot be established is which of the following:
    a. Correct with respect to its specification.
    b. Safe to operate.
    c. Robust in the presence of exceptional conditions.
    d. Considered to have passed verification.

5. In random ascent, we take the first neighbouring solution to show any improvement over the current solution as the new solution.
    a. True
    b. False

6. A mutant is considered valid, but not useful, if it compiles, but the majority of tests fail.
    a. True
    b. False

7. You are buying a server for your online business and are concerned with the recovery time after a failure. What reliability measure would be of most interest to you?
    a. Availability
    b. Probability of failure on demand
    c. Mean time between failures
    d. Rate of fault occurrence

Question 2 (Quality Scenarios) - 10 Points

Consider the cockpit display system for an aircraft. This system updates a display that offers the pilot important information on the status of the aircraft, including the fuel level, altitude, auto-pilot status, and any warnings about equipment failure or required aircraft maintenance.

Identify one performance and one availability requirement that you think would be necessary for this system and develop a quality scenario to verify each of them.

Write your own scenarios - do NOT reuse or lightly rewrite those from the slides or assignments.

Question 3 (Testing Concepts) - 9 Points

Choose one stage of testing (unit testing, system testing, GUI testing, exploratory testing, or acceptance 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 (System Testing) - 12 Points

Consider a feature that books a meeting in a designated room at a specified date, with a specified start and end time.

```
String bookMeeting (String roomID, String date, String startTime, String endTime)
```

This function returns a string with one of the following messages:

- “Booked” if the meeting was booked successfully.
- “Busy” if the room already has a meeting booked that overlaps with the requested timeframe.
- “Malformed Input” if any input items are malformed.
- “Illegal Time” if there are any errors related to the date, start time, or end time.

A well-formed date string is of the format “YYYY-MM-DD” (e.g., “2021-12-23”).
A well-formed time string is of the format “HH:MM” (e.g., “14:45”).

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

Perform category-partition testing for this feature.

1. For each parameter, identify testing choices (controllable items that can be varied when testing)
2. Identify representative input values (types of input choices) for each choice.
3. Apply constraints (IF, ERROR, SINGLE) where they make sense.

You do not need to create test specifications or concrete test cases.

Hint: Do not forget about environmental factors that can influence system output.

For invalid input, **do not** just write “invalid” - be specific.

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 a web-based inventory management system, where a user can (among other functionalities) check inventory status for a single item or all items, update the number of units of a particular item in the inventory, add new item types, and display and edit metadata about each item. Describe three actions you might take during exploratory testing of this system, based on the tour you described above. (You may make any assumptions you would like about the functionality of this system - as long as you state those exceptions clearly)

Question 6 (Unit Testing) - 8 Points

Consider a Java method for sorting an array:

```
int[] quickSort (int[] unsorted, int length);
```

Write JUnit-formatted test cases to do the following:

1. Check whether the unordered array given to the method is returned in the expected order.
2. Check whether an index out of bounds exception is thrown when the inputted length is either negative or greater than the length of the array.

Question 7 (Structural Testing) - 16 Points

For the following method:

```
1.  public static String collapseSpaces(String argStr){
2.      char last = argStr.charAt(0);
3.      StringBuffer argBuf = new StringBuffer();
4.      for(int cldx=0; cldx < argStr.length(); cldx++){
5.          char ch = argStr.charAt(cldx);
6.          if(ch != ' ' || last != ' '){
7.              argBuf.append(ch);
8.              last = ch;
9.          }
10.     }
11.     return argBuf.toString();
12.   }
```

1. Draw the control-flow graph for this method. You may refer to line numbers instead of writing the full code.
2. Develop test input that will provide statement coverage. For each test, list the statements covered.
3. Develop test input that will provide branch coverage. For each test, list the branches covered (use the line number and T or F, i.e., “3-T” for the true branch of line 3).
4. Develop test input that will provide basic condition coverage. For each test, list the conditions covered.

Question 8 (Data Flow Testing) - 12 Points

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

```
1. public String findLongestCommonSequence(String s1, String s2) {
2.     String result = "";
3.     for (int length = s1.length(); length > 0; length--) {
4.         int startIndex = 0;
5.         while (startIndex + length <= s1.length()) {
6.             String current = s1.substring(startIndex, startIndex + length);
7.             if (s2.contains(current)) {
8.                 result = current;
9.                 break;
10.             }
11.             startIndex++;
12.         }
13.         if (result.length() != 0) {
14.             break;
15.         }
16.     }
17.     return result;
18. }
```

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

Question 9 (Mutation Testing) - 15 Points

Consider the following function:

```
1.  int max (int a[], int n){
2.    int m = n - 1;
3.    while(n > 0){
4.      n = n - 1;
5.      if (a[n] > a[m]){
6.        m = n;
7.      }
8.    }
9.    return m;
10. }
```

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

- Relational operator replacement (ror)
- Arithmetic operator replacement (aor)
- 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.
3. For that mutant, provide test input that would detect the mutant. Show how the output (return value of the method) differs from that of the original program.