اختبار البرمجيات ٢٠١٦ - ٢٠١٧

19

Bar Code

Name

ID Number

Cairo University
Faculty of Computers and Information

**Course:** Software Testing (Selected Topics – 2)

**Course Code:** CS496

**Year:** Fall 2016-2017

**Date:** 4 Jan 2017

**Duration:** 2 hours

**Instructor:** Dr. Soha Makady

| |
|---|
| |
| 60 |

| Question | Mark |
|---|---|
| One | |
| Two | |
| Three | |
| Four | |
| Five | |
| Six | |
| Total Marks | |

**Total Marks in Writing:** ____________________

## Answer ALL Questions

**Question 1: (16 marks)**

a) Why would you need to apply the boundary value analysis technique, rather than relying only on the equivalence class partitioning technique? Give a simple example to support your answer. (2 marks)

b) Explain the difference between statement coverage and decision/branch coverage. (2 marks)

c) Define the terms verification and validation. (2 marks)

d) Give two examples of activities that are primarily used for verification and two examples of activities that are primarily used for validation. An activity can be, for example, unit test, integration test, system test, regression test, stress test, load test, acceptance test, usability test, requirements inspection, design inspection, code inspection, and so on. Briefly explain how each of the activities you picked supports verification or validation. (4 marks).

We have discussed several testing-related topics within your presentations.

e) Briefly explain what TMM stands for, and what its purpose is. (2 mark)

f) Within a testing team, several roles exist. Mention two of those roles, along with their responsibilities. (2 marks)

g) Within risk-based testing, several concepts were discussed including positive risks, and negative risks. Give an example for a positive risk, and an example for a negative risk within a project. (2 marks)

**<u>Question 2:</u> (8 marks)**

As explained during our lectures, every defined bug has a severity and a priority. You are reporting bugs on a bug tracking system called **MyBugPlanet. MyBugPlanet** provides four possible severity values:

- ***Sev. 1***: system crashes, security breach.
- ***Sev. 2***: wrong result, loss of functionality.
- ***Sev. 3***: Minor problem, misspelling, UI layout, rare occurrence.
- ***Sev. 4***: Suggestion.

Additionally, **MyBugPlanet** provides four possible priority values: ***Immediate fix***, ***Must fix***, ***Should fix***, and ***Would like to fix***. For each of the following bug descriptions, mention the suitable severity <u>and</u> priority. For each bug, you MUST justify your selections.

a) A data corruption bug that happens very rarely (e.g., that would change your balance in the bank to be 3 L.E. instead of 30 thousand L.E.).

b) A software release (for testing) that crashes on startup.

c) In a web application, the company's logo is not displayed properly.

d) In a social security insurance registration portal (موقع لتسجيل المواطنين للتأمين الاجتماعي), the user cannot update his existing telephone number if it changes, in case the government needs to contact him. But, the government can still contact the users by sending letters to their home addresses.

**Question 3:** (8 marks)

Consider a Paragraph Effects configuration window, which allows the user to update three different paragraph settings as follows:

Paragraph spacing: Takes one of two possible values (Allow adding spaces between paragraphs or Disallow adding spaces between paragraphs).

Indentation: Takes one of two values (Allow mirroring indents or Disallow mirroring indents).

Line spacing: Takes one of three values (Single line spacing, Double line spacing, or Multiple line spacing).

a) What is the largest number of possible test cases needed to exhaustively test such a window? You need to explain your calculation that resulted in your answer.

b) Apply pair-wise testing technique to reduce the number of test cases as much as possible. You need to show how many test configurations would you have. You need to show your calculations, and to list all the tests that you will have after applying such a technique. You also need to show how you derived those tests.

**Question 4: (8 Marks)**

A warehouse software system is intended to accept the name of a soft drink item, dname, and a list of the different sizes the item comes in, gsize1 to gsize5, specified in litres. A maximum of five sizes may be entered for each item. The specification states that the item name is to be entered first, followed by a comma, then followed by a list of sizes. A comma must be used to separate each size. Spaces (blanks) are to be ignored by the software anywhere in the input. The input must be entered in one line. The item name is to be alphabetic characters with a length of 2 to 15 characters. Each size may take a value in the range of 1 to 10, whole numbers (integers) only. The sizes are to be entered in ascending order (smaller sizes first).

Based on this specification, many Equivalence Classes for testing the software module can be derived. The following list shows some examples:

1. Item name is alphabetic (valid)
2. Item name is not alphabetic (invalid)
3. Size value is less than 1 (invalid)
4. Size value is in the range 1 to 10 (valid)

Add 6 additional equivalence classes (including at least two invalid ones), and provide a set of test cases that cover the 10 classes listed. Try to make the set of test cases as small as possible. Remember that a complete test case also contains an output value. To make things easy, assume that there are only two output values, i.e., 'input accepted' and 'invalid input', for valid and invalid inputs, respectively. For each test case, please mention the equivalence classes that it covers.

**Question 5:** (10 Marks)

Consider the following Java class listing that finds the index of the maximum value in an array:

```
1   public class FindMaxIndex {
2       static int indexer(int[ ]  a) {
3           int i =0;
4           int index = 0;
5           while (i< a.length - 1)
6           {   i = i + 1;
7               if (a [i] > a[index])
8                   index = i;
9           }
10          return index;
11      }
12  }
```

The following test suite (including three test cases) has already been developed by applying black-box testing techniques for this program:

T = {

Test case 1:<a[0] = 0, a[1] = 100, a[2] = 200>, expected behavior: Index of maximum value = 2,

Test case 2:<a[0] = 0, a[1] = 200, a[2] = 100>, expected behavior: Index of maximum value = 1,

Test case 3:<a[0] = 200, a[1] = 0, a[2] = 100>, expected behavior: Index of maximum value = 0,

}

Perform the mutation testing/analysis process on this program for the following three mutants. Note that each mutant is applied on the original program, and **NOT** on the previous mutants. Make sure you explain clearly in each stage if the mutant under analysis is killed or not. If the mutant is not killed, what test case should be added to kill it. Also, calculate the mutation score in each stage.

- Mutant 1: replace line 7 by: `if (a [i] >= a[index])`
- Mutant 2: replace line 4 by: `int index = a.length – 1;`
- Mutant 3: replace line 8 by: `index = i-1;`