SENG 275 MID TERM EXAM 1 Solution-O2 (10%)

Instructor: Dr. Navneet Kaur Popli, Date: 30 May 2023, Time: 12:30-1:20 PM PST
Mode: Pen-Paper, Synchronous, Timed, Closed-book
Total Marks: 44, No. of pages: 7, No. of questions: 18
Note: This is an individual activity. Copying or cheating of any kind is not allowed. Supplement with diagrams, tables, graphs, and charts wherever applicable.

Q1) Having certain terminology helps testers to explain the problems they have with a program or in their software.(3M)

Below is a small conversation. Fill in the blanks with one of the following terms: failure, fault, or error.

- **Mark**: Hey, Jane, I just observed a _ _ _ _ _ _ in our software: if the user has multiple surnames, our software doesn't allow them to sign in.
- **Jane**: Oh, that's awful. Let me check the code so that I can find the _ _ _ _ _ _.
- **Jane** *(a few minutes later):* Mark, I found it! It was an _ _ _ _ _ _. I programmed that part, but never thought of this case. I should be paying more attention to the documentation.
- **Mark**: No worries, Jane! Thanks for fixing it!

A1)

1. Failure: the user notices the system/program behaving incorrectly.
2. Fault/bug/defect: this is a problem in the code, that is causing a failure in this case.
3. Error/mistake: the human mistake that created the fault.

Q2) A tester was doing a code review for a SUT. They came across the following code: (4M)
(a) Identify the fault in this piece of code.
(b) If possible, identify a test case that does not execute the fault.
(c) If possible, identify a test case that executes the fault, but does not result in an error state.
(d) Fix the fault and verify that the given test now produces the expected output.

```
public static int oddOrPos(int[] x)
{
//Effects: if x==null throw NullPointerException
// else return the number of elements in x that
//      are either odd or positive (or both)
   int count = 0;
   for (int i = 0; i < x.length; i++)
   {
      if (x[i]%2 == 1 || x[i] > 0)
      {
         count++;
      }
   }
   return count;
}
   // test:  x=[-3, -2, 0, 1, 4]
   //        Expected = 3
```

(a) The if-test needs to take account of negative values (positive odd numbers are taken care of by the second test). So the condition x[i]%2= =1 is not required.
if (x[i]%2 = = -1 || x[i] > 0)
Working of modulo function in Java:
-11 % 5 == -1
11 % -5 == 1
-11 % -5 == -1
The sign of the first operand decides the sign of the result.
`x % y` always equals `x % -y`.
You can think of the sign of the second operand as being ignored.

(b) x must be either null or empty. All other inputs result in the fault being executed. We give the empty case here.
Input: x = []
Expected Output: 0
Actual Output: 0

(c) Any nonempty x with only non-negative elements works, because the first part of the compound if-test is not necessary unless the value is negative.
Input: x = [1, 2, 3]
Expected Output: 3
Actual Output: 3
(d) see (a)

Q3) Which of the following are valid testing principles? (1M)
I) Exhaustive testing is in general impossible.
II) Exhaustive testing should be executed for code intended to be reused.
III) Testing may guarantee that a program is correct.
IV) Testing cannot guarantee that a program is correct.

V) Defects cluster together in certain areas of the product.

a) I, IV, V
b) I, V
c) I, III
d) II, IV

Q4) Which of the following could be good examples of what you might test in unit testing (select all that apply). (2M)

a) Multiple classes for feature implementation
b) A whole class
c) A single method
d) A functional use case implementation

A4) all are correct. 0.5 mark for each

Q5) Match the **testing principles** (from the list shown on the right) that **best** describes the concepts or examples shown on the left. (5M)

➡ __4__ Sally is a new developer at the University of Victoria and she notices that many of the complaints about Brightspace all seem to be about the Quiz module! Which testing principle best describes this phenomenon.

➡ __5__ Testing your program to make sure that it conforms to the user requirements and the expected user inputs is often the first goal when writing test cases before writing or designing your code. Which testing principle best describes this situation.

➡ __2__ Peter is an experienced developer and believes that he can fully test Microsoft Teams so that it never crashes! Which testing principle best explains why this is **NOT** possible?

➡ __3__ Some years ago Google developed a communication tool called "Google Wave" that Googlers loved and it worked flawlessly. However, most people did not find it useful for their needs. Which testing principle best describes this problem?

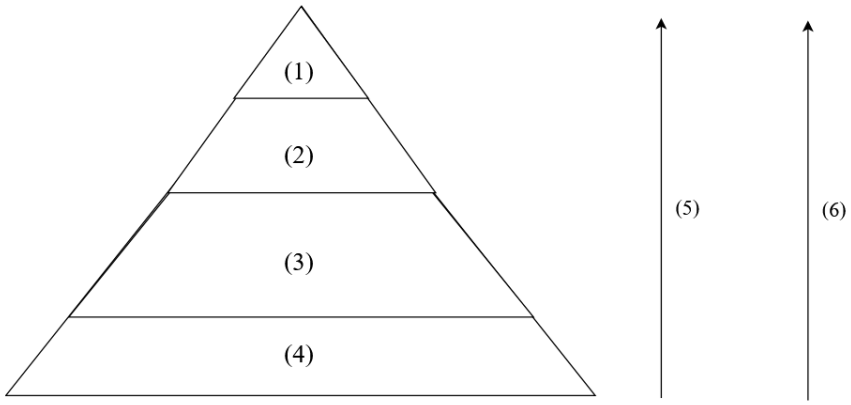
1. Pesticide Paradox
2. Exhaustive Testing is Impossible
3. Absence of errors is a fallacy
4. Defect Clustering
5. Good Weather Testing

➡ __1_ Jean believes that unit testing is the most effective way to use automated testing so they put all their energy into that type of testing. Which testing principle captures that this is not the preferred approach!

Q6) What do you understand by scaffolding? (1M)
A6) How do we run a module without anything to call it, to be called by it or, possibly, to output intermediate values obtained during execution?
One approach is to construct an appropriate driver routine to call it and, simple stubs to be called by it, and to insert output statements in it. A stub or dummy subprogram uses the subordinate module's interface.
This overhead code, called scaffolding represents effort that is important to testing, but does not appear in the delivered product. Used normally in integration testing but sometimes can be used in unit testing as well.

Q7) System testing is a type of white box-testing. (True/False)

Q8) Here is a skeleton for the testing pyramid. Fill in the correct corresponding terms.



A8)

1. Manual
2. System
3. Integration
4. Unit
5. More reality (interchangeable with 6)
6. More complexity (interchangeable with 5)

Q9) A newly developed product started off with some basic unit tests but later on decided to add only integration and system tests for the new code that was written. This was because a user interacts with the system as a whole and therefore these types of tests were considered more valuable. Therefore, unit tests became less prevalent, while integration and system tests became a more crucial part of the test suite. Which of the following describes this transition? (1M)

1. Transitioning from a testing pyramid to an ice-cream cone anti-pattern
2. Transitioning from an ice-cream cone anti-pattern to a testing pyramid
3. Transitioning form an ice-cream cone pattern to a testing pyramid

Q10) A program called “Software-Testing” does the following:

Given an integer ‘n’, return the string formed from the number n (using the logic explained below) followed by “@”.

- If the number is divisible by 2, use “Software” instead of the number; and
- if the number is divisible by 7, use “Testing” instead of the number, and
- if the number is divisible by both 2 and 7, use “Software-Testing” instead of the number.
- if the number is neither divisible by 2 or 7, use “n@” instead of the number ‘n’.

Examples:

- The integer 22 yields “Software@”
- The integer 77 yields “Testing@”
- The integer 42 yields “Software-Testing@”

- The integer 201 yields “201@”

A novice tester is trying to devise as many tests as possible for the ‘Software-Testing’ program and comes up with the following:

- T1 = 10
- T2 = 28
- T3 = 14
- T4 = 99
- T5 = 12
- T6=35
- T7=21
- T8=23

Which of these tests can be removed while maintaining a good test suite and why? Explain in detail. (3M)
Which concept can we use to determine the test(s) that can be removed? (1M)

A10)

Partitions are:

P1: n%2
P2: n%7
P3: n%2 and n%7
P4: n not divisible by either 2 or 7

T1,T5: P1 so one of them can be removed.
T7, T6: P2 so one of them can be removed.
T3,T2:P3 so one of them can be removed.
T8,T4:P4 so one of them can be removed.

Equivalence partitioning concept is used here.

Q11) Choosing the level of a test involves a trade-off. After all, each test level has advantages and disadvantages. Which one of the following is the **main advantage** of a test at system level?

1. The interaction with the system is much closer to reality.
2. In a continuous integration environment, system tests provide real and quick feedback to developers.
3. Given that system tests are never flaky, they provide developers with more stable feedback.
4. A system test is written by product owners, making it making it less complex and closer to reality.

Q12) What is the main reason for the number of recommended system tests in the testing pyramid to be smaller than the number of unit tests? (1M)

1. Unit tests are as good as system tests.
2. System tests do not provide developers with enough quality feedback.
3. There are no good tools for system tests.
4. System tests tend to be slow and are difficult to make deterministic.

Q13) Differentiate between smoke and sanity testing. (2Marks)
A13) Testing the build for the very first time is to accept or reject the build. This we call it as Smoke Testing. If the test team accepts the build, then that particular build goes for further testing. Imagine the build has 3 modules namely Login, Admin, Employee. The test team tests the main functionalities of the application without going deeper. This we call it as Sanity Testing. Once sanity tests pass further testing on the build can be done for additional functionalities.

Q14) A software organization can be sued for damages if their application crashes and causes financial harm to the user. Given below is one way the organization can help pay legal costs: (Choose one) (1Mark)
a) Create careful End User Licence Agreements which stand their ground in court
b) Take Errors and Omissions insurance
c) Take Software Development and Testing Insurance
d) Show due diligence during development and testing

Q15) For someone to successfully sue a software developer for negligence in absence of a contract, they must prove these 4 things (select all 4 that apply) :

a) Duty of Care
b) Duty to warn
c) Standard of Care
d) Failure to warn
e) Damages/Injury
f) Failure to perform
g) Shared liability
h) Causation

Q16) Explain using any 2 examples how you can find errors in the requirements stage. (2M)
A16)We can find if requirements are ambiguous, eg. website should load quickly is an ambiguous requirement. Instead saying that website should load in maximum of 0.5 seconds is a more non-ambiguous requirement.
Similarly say a person says I want red colour of the home page, you can specifically prototype the colour and give it a specific RGB value.
Prototyping, mocking, scenario techniques could be used.

Q17) Code reviews are an example of validation. (True/False) (1M)

Q18) What are flaky tests? Explain any one way to spot a flaky test and any one way to handle them once you have spotted them. (3M)

A18) A flaky test is a test that presents an erratic behavior: if you run it, it might pass or it might fail for the same configuration.
You can retry a test at least 3 times to spot a flaky one. You can also change order of tests to determine flakiness.
You can quarantine flaky tests into their own test suite once you have spotted them and run the suite separately from the other test suites. Check for memory leaks.