United International University
Department of Computer Science and Engineering

CSE 4495: Software Testing and Quality Assurance
Final Examination : Summer 2023
Total Marks: 40 Time: 2 hours

Any examinee found adopting unfair means will be expelled from the trimester / program as per UIU disciplinary rules.

Answer all the questions. Numbers to the right of the questions denote their marks.

1. (a) You are tasked with solving an issue regarding the recently deployed *Requisition Digitization* system at UIU for faculty members and staff. The issue states that although the system properly worked in the first testing phase with 3-4 sample users, after deployment the server is frequently facing crashes and request timeouts. Moreover, extreme variation in response time is perceived when many users are concurrently using the system. Determine the quality attribute, the lack of which is causing this issue. What are the possible ways to solve this if new hardware resources can be acquired? [2]

   (b) Imagine you are the quality assurance lead of an online shopping service. While testing your company's upcoming mobile application you were provided the following requirements for product release- availability of at least 99.9% and *ROCOF* less than 9 errors per 24 hour period [4]
   After the testing is done you receive the following report from your testing team - "During 10 days of testing the system processed 7560 requests.

   i. After each crash servers were restarted. Each restart took 30 minutes on average."
   ii. 97.5% of all requests were successful.
   iii. 65.1% of failed requests ended up causing a system crash.

   Now depending on this report measure the availability and *ROCOF,POFOD* of your system. Also decide whether your product meets the requirements given for release. Also Can you calculate the MTBF?

2. (a) For this year's CSE project show you want to develop a website named *CodeWars* for UIU CSE students where they can host their own private programming contests among UIU students. In addition, students can practice from a large collection of problems, add other users as friends, throw out challenges to each other, etc. Now, On this website, you want to test the search and sort functions. The functionality of this feature is that it takes an input keyword from the user and displays a list of all problems that contain the keyword in their title. It also displays the problems in ascending order of difficulty. To test this feature which types of oracle(s) should a tester use? describe the design and purpose of these oracle(s). [2]

   (b) Consider the scenario described in 2.(a), As you proposed the idea to your project group mates, one of your teammates suggested that you complete the whole development phase at once and test the highest-level APIs only since writing tests for low-level functionalities is tedious and time-consuming while another suggested that you should conduct the testing process parallelly with each phase of the development and write independent test suits for each class, then each module and after that each high-level feature. Which idea is more aligned with industry best practices? State three reasons in support of your choice. (Do not write unnecessary and vague reasons, as you will get no marks) [2]

   (c) Briefly describe the different stages of testing your second teammate suggested to implement. [2]

3. (a) The following ***AddToCart*** is a high-level feature exposed by the API of an e-commerce platform. The functionality is pretty simple- if a product is in stock it can be added to the cart, if the product is already in the cart the system increases the quantity by one. So, basically *product* is a string and the cart is a product-quantity hashmap that stores the product names and corresponding quantity of each product. [5]

   $$AddtoCart(String\ product, Hashmap\langle String, int\rangle\ cart)\ returns\ updatedCart$$

   You need to design system-level test cases using the category-partition method for the *AddToCart* function. So identify choices(aspects that you control and that can vary the outcome), a set of representative values for each choice, and find out the total number of test specifications for your choices.[Hint: the product database should be considered as a choice.]

   (b) Apply necessary **ERROR**, **SINGLE** and at least 1(one) **IF** constraints to your representative values derived in Q.3(a). Calculate the reduced number of test specifications generated after applying these constraints. [4]

   (c) consider the following security and privacy settings choices for chromium-browser:
   Create a covering set for pair-wise combinatorial testing of these browser settings. [3]

Allow Third-party cookies	Browsing mode	Use Secure DNS
Allow	Standard Protection	Use
Restricted	Enhanced Protection	Don't Use
Block	No Protection	

4. You need to write unit tests for a personal task scheduler app you are developing. Consider the following class diagrams in Fig.1 for your app. Each task is created with a name and a priority level (1 = HIGH, 2=MEDIUM, 3 = LOW). If a task is added to the schedule it will be inserted according to the priority ($HIGH > MEDIUM > LOW$). For example, if there are two tasks in the schedule currently, one with HIGH priority and one with LOW: The HIGH priority will have position 1 and the LOW one will be at position 2. And, if you add another task with MEDIUM priority it will automatically be inserted between the two previous tasks. If tasks with the same priority are added they will appear in the order in which they were inserted. Write JUnit executable unit tests for the following scenarios.

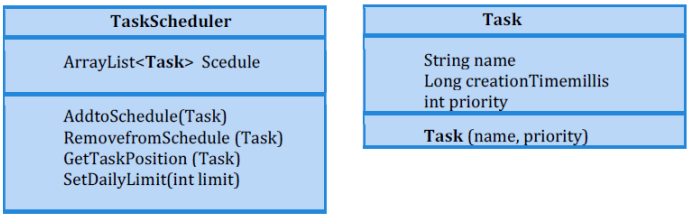

Figure 1: Class diagrams for Ques-4

(a) Test the *AddtoSchedule(Task t)* method by consecutively creating tasks with priority in the following order - **low, low, high, medium**, and adding them. After adding each task you must check whether it has been added in the appropriate position by calling the *GetTaskPosition(Task)* for that task. [4]

(b) Test the *SetDailyLimit(int limit)* method which sets a limit on how many tasks can be added. If you attempt to add more tasks than the limit the system should through a *ScheduleFull* exception with the message *"Your schedule is completely full. Increase daily limit to add more tasks."* [4]

5. Consider the following code:

```
1. public int doSomething(int x, int y)
2.  {   while(y > 0) {
3.          if(x > 0) {
4.              y = y - x;
5.                  if (y > 0)
6.                     System.out.println("Y: " + y);
7.           }else {
8.              x = x + 1;
9.                 if (x <= 0)
10.                    System.out.println("X: " + x);
11.         }
12.     }
13.     return x + y;
14. }
```

(a) Draw the control flow graph for the function ***doSomething***. [3]

(b) Trace the control flow path in your CFG for the following test inputs: [2]
$doSomething(10, -1)$, $doSomething(3, 4)$, $doSomething(1, -1)$

(c) Define fault-based testing. What is the difference between first and second-order mutants? [2]

(d) Briefly describe the purpose and workflow of ***one*** testing tool from the list below: [1]

- PyTest
- OWASP-ZAP
- Cypress
- Apache Jmeter