DEPARTMENT OF COMPUTER SCIENCE
MILITARY COLLEGE OF SIGNALS, NUST
SOFTWARE QUALITY ASSURANCE
BESE-14 A&B

Exam: Final Term
Type of Paper: Regular
Semester: Fall

Instructor: Dr. Seemab Latif
Maximum Marks: 40
Time Allowed: 2.5 hours

Note:

1. This question paper has 2 pages and 4 questions.
2. Attempt all questions.
3. Use of programmable calculators with memory is not allowed

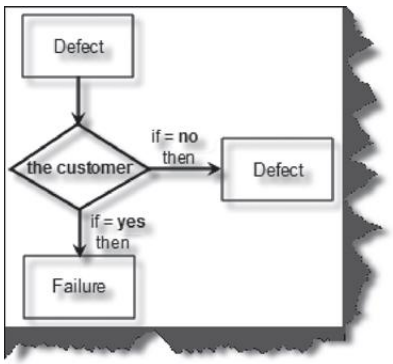
Question No.1: **(10)**

a. Briefly describe difference between a Run chart and Control Chart. How both can help us in SQA?

A run chart tracks the performance of the parameter of interest over time. A control chart can be regarded as an advanced form of a run chart for situations where the process capability can be defined. A run chart is best used for trend analysis, especially if historical data are available for comparisons with the current trend. A control chart is used to assess the stability of the product.

b. What is the difference between a defect and a failure?

When a defect reaches the end customer it is called a failure and if the defect is detected internally and resolved it's called a defect.



c. Briefly describe difference between a Pareto chart and Histogram. How both can help us in SQA?

A Pareto diagram is a frequency chart of bars in descending order; the frequency bars are usually associated with types of problems. The histogram is a graphic representation of frequency counts of a sample or a population. By arranging the causes based on defect frequency, a Pareto diagram can identify the few causes that account for the majority of defects. It indicates which problems should be solved first in eliminating defects and improving the operation. In a histogram, the frequency bars are shown by the order of the X variable, whereas in a Pareto diagram the frequency bars are shown by order of the frequency counts. The purpose of the histogram is to show the distribution

characteristics of a parameter such as overall shape, central tendency, dispersion, and skewness. It enhances understanding of the parameter of interest.

d. Briefly outline the difference between masked and latent defect.

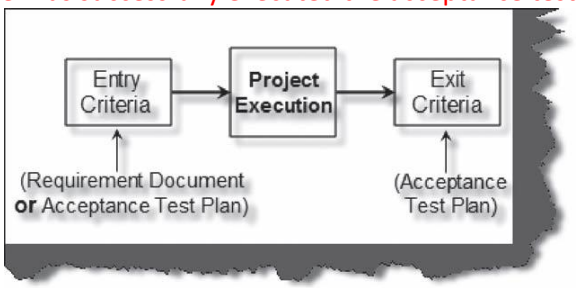
A latent defect is an existing defect that has not yet caused a failure because the exact set of conditions was never met. A masked defect is an existing defect that hasn't yet caused a failure just because another defect has prevented that part of the code from being executed.

e. Perception of Quality or Quality view is same among people and organizations. True/False, briefly explain.

False, quality has different views, transcendental view, product view, manufacturer view, user view and value-based view**.**

f. What does entry and exit criteria mean in a project?

You can define entry criteria that the customer should provide the requirement document or acceptance plan. If this entry criterion is not met then you will not start the project. On the other end, you can also define exit criteria for your project. For instance, one of the common exit criteria in projects is that the customer has successfully executed the acceptance test plan.



g. How fish bone diagramming method is helpful?

It shows the relationship between a quality characteristic and factors that affect that characteristic. Fish bone diagram identifies all causal factors of a quality characteristic in one chart.

h. Name different test plan documents in a project life cycle?

There are four test plan documents.

- Acceptance test plan
- System test plan
- Integration testing
- Unit testing

i. Which test cases are written first: white boxes or black boxes?

Normally black box test cases are written first and white box test cases later.

j. What's the difference between System Testing and Acceptance Testing?

Acceptance testing checks the system against the "Requirements." It is similar to System testing in that the whole system is checked but the important difference is the change in focus: System testing checks that the system that was specified has been delivered. Acceptance testing checks that the system will deliver what was requested. The customer should always do Acceptance testing and not the developer.

Question No.2: (Unit Testing) **(4+2)**

a. Draw a data flow graph for the binsearch() function given in Figure 1.

```
int binsearch(int X, int V[], int n){
    int low, high, mid;
    low = 0;
    high = n - 1;
    while (low <= high) {
      mid = (low + high)/2;
      if (X < V[mid])
          high = mid - 1;
      else if (X > V[mid])
           low = mid + 1;
      else
           return mid;
    }
    return -1;
}
```

Figure 1: Binary Search Function

b. By referring to the data flow graph obtained in part a, define-use-kill patterns for variable mid (taken in pairs as the paths are followed).

Path 1: du,uu,uu
Path 2: du,uu,uu
Path 3: du,uu,uu
Path 4: du,uu
Path 5: ~d

Question No.3: (Integration Testing) **(2+2+2)**

a. Discuss the advantages and disadvantages of top-down and bottom-up approaches to integration testing.

The limitations of the top-down approach are as follows:

a. Until a certain set of modules has been integrated, it may not be possible to observe meaningful system functions because of an absence of lower level modules and the presence of stubs.
b. Test case selection and stub design become increasingly difficult when stubs lie far away from the top-level module. This is because stubs support limited behaviour, and any test run at the top level must be constrained to exercise the limited behaviour of lower level stubs.

The advantages of the top-down approach are as follows:

a. System integration test (SIT) engineers continually observe system-level functions as the integration process continues. How soon such functions are observed depends upon their choice of the order in which modules are integrated. Early observation of system functions is important because it gives them better confidence.
b. Isolation of interface errors becomes easier because of the incremental nature of top-down integration.
c. Since test inputs are applied to the top-level module, it is natural that those test cases correspond to system functions, and it is easier to design those test cases than test cases designed to check internal system functions. Those test cases can be reused while performing the more rigorous, system-level tests.

The disadvantages of the bottom-up approach are as follows:

a. Test engineers cannot observe system-level functions from a partly integrated system. In fact, they cannot observe system-level functions until the top-level test driver is in place.
b. Generally, major design decisions are embodied in top-level modules, whereas most of the low-level modules largely perform commonly known input–output functions. Discovery of major flaws in system design may not be possible until the top-level modules have been integrated.

The advantages of the bottom-up approach are as follows.

a. If the low-level modules and their combined functions are often invoked by other modules, then it is more useful to test them first so that meaningful effective integration of other modules can be done. In the absence of such a strategy, the testers write stubs to emulate the commonly invoked low-level modules, which will provide only a limited test capability of the interfaces.

a. Suppose that you plan to purchase COTS components and integrate them with your communication software project. What kind of acceptance criteria will you develop to conduct acceptance testing of the COTS components?

1. Functional Correctness and Completeness
2. Accuracy
3. Data integrity
4. Data conversion
5. Data and recovery

6. Competitive edge
7. Usability
8. Performance
9. Startup time
10. Stress
11. Availability and reliability
12. Robustness
13. Maintainability and serviceability
14. Interoperability
15. Scalability
16. Documentation

b. If a program passes all the black-box tests, it means that the program should work properly. Then, in addition to black-box testing, why do you need to perform white-box testing?

a. When using black box testing, the tester can never be sure of how much of the SUT has been tested. No matter how clever or diligent the tester, some execution paths may never be exercised.
b. To find every defect using black box testing, the tester would have to create every possible combination of input data, both valid and invalid. This exhaustive input testing is almost always impossible. We can only choose a subset (often a very small subset) of the input combinations.
c. When using white box testing, the tester can be sure that every path through the software under test has been identified and tested.

Question No.4: (System Testing) (2+2+2)

a. What is an element management system (EMS)? How is it different from a network management station (NMS)?

The EMS tests verify the main functions which are to manage, monitor, and upgrade the communication system network elements (NEs). An EMS communicates with its NEs by using, for example, the Simple Network Management Protocol (SNMP) and a variety of proprietary protocols and mechanisms.

A Network Management System (NMS) is a combination of hardware and software used to monitor and administer a network. Individual network elements (NEs) in a network are managed by an element management system.

b. Discuss the importance of regression testing when developing a new software release. What test cases from the test suite would be more useful in performing a regression test?

In this category, new tests are not designed. Instead, test cases are selected from the existing pool and executed to ensure that nothing is broken in the new version of the software. The main idea in regression testing is to verify that no defect has been introduced into the unchanged portion of a system due to changes made elsewhere in the

system. During system testing, many defects are revealed and the code is modified to fix those defects.

Regression testing is an expensive task; a subset of the test cases is carefully selected from the existing test suite to (i) maximize the likelihood of uncovering new defects and (ii) reduce the cost of testing.

c. What are the differences between performance, stress, and scalability testing? What are the differences between load testing and stress testing?

Performance tests measure the performance characteristics of the system, for example, throughput and response time, under various conditions. Stress tests stress the system in order to determine the limitations of the system and determine the manner in which failures occur if the system fails. Scalability tests determine the scaling limits of the system.

Difference between Load and Stress testing: In load testing, the objective is to ensure that the system can operate on a large scale for several months, whereas, in stress testing, the objective is to break the system by overloading it to observe the locations and causes of failures.

Question No.5 (Acceptance Testing) (1+2+5)

a. What are the objectives of acceptance testing?

It is unlikely that the system passes all the acceptance criteria in one go for large, complex systems. It is useful to focus on the following three major objectives of acceptance testing for pragmatic reasons:

a. Confirm that the system meets the agreed-upon criteria.
b. Identify and resolve discrepancies, if there are any.
c. Determine the readiness of the system for cut-over to live operations. The final acceptance of a system for deployment is conditioned upon the out-come of the acceptance testing. The acceptance test team produces an acceptance test report which outlines the acceptance conditions.

b. Who should define the acceptance quality attribute criteria of a test project? Justify your answer?

Formulation of acceptances criteria is governed by the business goals of the customer's organization.

The customer needs to select a subset of the quality attributes and prioritize them to suit their specific situation. Next, the customer identifies the acceptance criteria for each of the selected quality attributes. When the customer and the software vendor reach an agreement on the acceptance criteria, both parties must keep in mind that satisfaction of the acceptance criteria is a trade-off between time, cost, and quality.

c. In the following series of application examples, provide four most important quality attributes discussed in the lecture that you think are critical to the applications.

a. Telecommunication software
   Functional Correctness, data conversion, scalability and performance

b. Video game software

Performance, scalability, interoperability, competitive edge

c. Intrusion detection system (IDS) software
Competitive edge, Functional correctness, performance, reliability

d. Medical diagnostic software for assisting doctors
Functional correctness, accuracy, performance, reliability

e. Highly popular music download website
Competitive edge, performance, availability, stress, scalability

f. Operating system on deep space mission probe
Functional correctness, accuracy, performance, reliability, documentation

g. Pension payment calculation system
Accuracy, functional correctness, data conversion, competitive edge

h. E-mail system
Stress, scalability, reliability, data integrity

i. Aircraft system software
Functional correctness, performance, availability, stress, documentation

j. Semi-automated telephone service
Functional correctness, accuracy, performance, reliability, stress, scalability

Question No.6 (Software Reliability) **(2+3)**

a. What are the factors that influence software reliability? Describe briefly.

1 **Size and Complexity of Code:** The number of LOC in a software system is a measure of its size. Large software systems with hundreds of thousands of LOC tend to have more faults than smaller systems. The likelihood of faults in a large system, because of more module interfaces, is higher. The more number of conditional statements the code contains, the more complex the system is considered to be. Due to the economic considerations in software development, one may not have much time to completely understand a large system. Consequently, faults are introduced into the system in all its development phases. Similarly, while removing faults from a large system, new faults may be introduced.

2 **Characteristics of Development Process:** Much progress has been made in the past few decades in the field of software engineering. New techniques and tools have been developed to capture system requirements, to design software systems, to implement designs, and to test systems.

3 **Education, Experience, and Training of Personnel:** The information technology industry has seen tremendous growth in the past 20 years or so. It is not unusual to

find many personnel with little training to be writing, modifying, and testing code in large projects. Lack of desired skills in personnel can cause a system to have a larger number of faults.

4 **Operational Environment:** Detection of faults remaining in a software system depends upon a test engineer's ability to execute the system in its actual operational environment. If a test engineer fails to operate a system in the same manner the users will do, it is very likely that faults will go undetected. Therefore, test engineers must understand the ways the users will operate a system. Because of a lack of sufficient time and lack of experience on the part of development and test engineers, faults can be introduced into a system, and those faults can go undetected during testing.

b. What definition of software reliability is best suited for each of the following kinds of software systems? Justify your answers.

Definition 1: Software reliability is defined as the probability of failure-free operation of a software system for a specified time in a specified environment. The key elements of the above definition of reliability are as follows: Probability of failure-free operation, Length of time of failure-free operation, A given execution environment. Software reliability is expressed as a continuous random variable due to the fact that most large software systems do have some unknown number of faults, and they can fail anytime depending upon their execution pattern. Therefore, it is useful to represent software reliability as a probabilistic quantity.

Definition 2: Failure intensity is a measure of the reliability of a software system operating in a given environment. According to the second definition, the lower the failure intensity of a software system, the higher is its reliability. To represent the current reliability level of a software system, one simply states the failure intensity of the system. For example, let a software system be in its system testing phase where test engineers are observing failures at the rate of 2 failures per eight hours of system execution. Then, one can state the current level of the reliability of the system as 0.25 failure per hour.

1 **Operating system of computer for personal use**
Failure intensity

2 **Operating system of computer used to control electric power plant**
Probability of failure-free operation

3 **Spell checking system**
Failure intensity

4 **Embedded software controlling washing machine**
Probability of failure-free operation

5 **Communication software system running on cellular phone**

Probability of failure-free operation

6 **Communication software system controlling base stations of cellular phone system**
Probability of failure-free operation

Extra Credit: **(1)**

ATP stands for:

- **Avocado Tree Plant**
- **Acceptance Test Plan**
- **Automatic Telecommunication Phase**
- **Automatic Text Pasting**