



MODEL ANSWER
SUMMER- 17 EXAMINATION

Subject Title: **SOFTWARE TESTING**

Subject Code:

17624

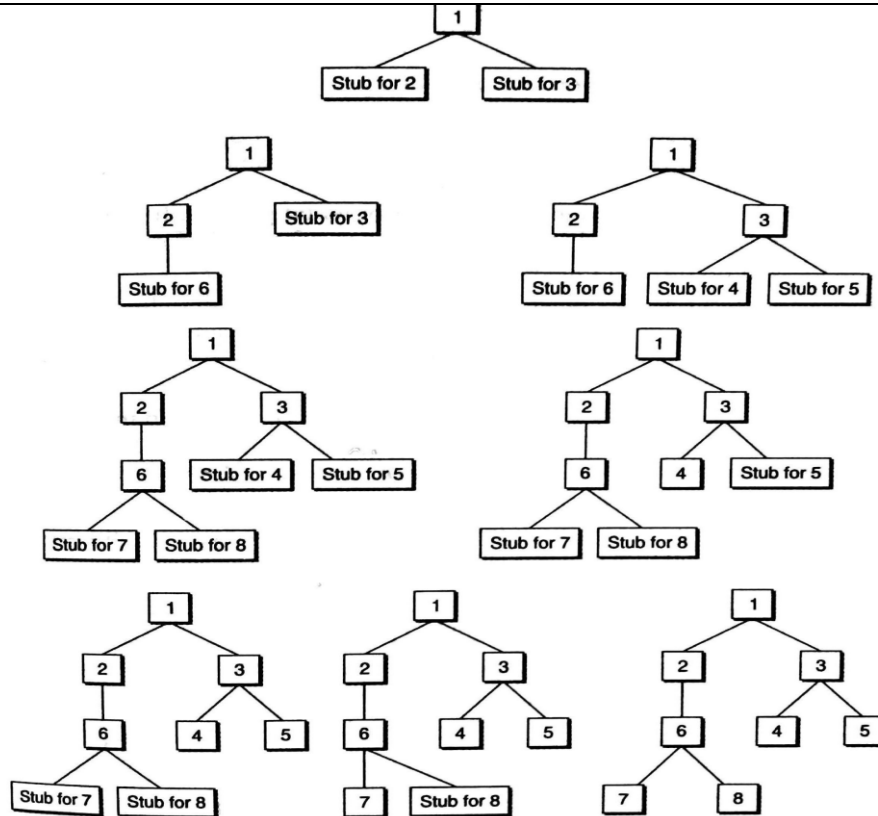
Important Instructions to examiners:

- 1) The answers should be examined by key words and not as word-to-word as given in the model answer scheme.
- 2) The model answer and the answer written by candidate may vary but the examiner may try to assess the understanding level of the candidate.
- 3) The language errors such as grammatical, spelling errors should not be given more Importance (Not applicable for subject English and Communication Skills).
- 4) While assessing figures, examiner may give credit for principal components indicated in the figure. The figures drawn by candidate and model answer may vary. The examiner may give credit for any equivalent figure drawn.
- 5) Credits may be given step wise for numerical problems. In some cases, the assumed constant values may vary and there may be some difference in the candidate's answers and model answer.
- 6) In case of some questions credit may be given by judgement on part of examiner of relevant answer based on candidate's understanding.
- 7) For programming language papers, credit may be given to any other program based on equivalent concept.

Q. No	Sub Q. N.	Answer	Marking Scheme
1.	(A)	Attempt any THREE of the following :	12 Marks
	(a)	Define software testing. List all skills of software tester.	4M
	Ans:	<p>Software Testing is the process of executing a program with the intent of finding errors. A successful test is one that uncovers an as-yet-undiscovered error. Testing can show the presence of bugs but never their absence. Testing is a support function that helps developers look good by finding their mistakes before anyone else does. Execution of a work product with intent to find a defect. Prevents defects.</p> <p>Different Skills of software tester :</p> <ul style="list-style-type: none"> ● Communication skills ● Domain knowledge ● Desire to learn ● Technical skills ● Analytical skills ● Planning ● Integrity ● Curiosity ● Think from users perspective ● Be a good judge for product 	(Definition: 2 marks, List of Skills of Software tester: 2 marks)
	(b)	Describe Inspection under static testing.	4M
	Ans:	Under Static testing is to review the code without executing it. Inspection is the most formal method in static testing. This method can detect all faults, violations and other side effects.	(Explanation: 4 marks)



	<p style="text-align: center;">OR</p> <p>Inspection is formal review where people external to the testing team may be involved as inspectors. They are subject matter experts who review the work product.</p> <p>In this:</p> <ol style="list-style-type: none"> 1. Thorough preparation is required before an inspection/review 2. Enlisting multiple diverse views. 3. Assigning specific roles to the multiple participants 4. Going sequentially through the code in a structured manner. <p>There are four roles in inspection:</p> <ol style="list-style-type: none"> 1. Author of the code: the person who had written the code 2. Moderator: who is expected to formally run the inspection according to the process? 3. Inspectors: are the people who actually provide review comments for the code. 4. Scribe: who takes detail notes during the inspection meeting and circulates them to the inspection team after the meeting. <p>The author or moderator selects review team. The inspection team assembles at the agreed time for inspection meeting. The moderator takes the team sequentially through the program code. If any defect is found they will classify it as minor or major. A scribe documents the defects. For major defects the review team meets again to check whether the bugs are resolved or not.</p>	
	<p>(c) Describe Top-Down integration testing with labelled diagram.</p>	<p>4M</p>
<p>Ans:</p>	<p>The strategy in top-down integration is look at the design hierarchy from top to bottom. Start with the high - level modules and move downward through the design hierarchy. Modules subordinate to the top modules are integrated in the following two ways:</p> <ol style="list-style-type: none"> 1. Depth first Integration: In this type, all modules on major control path of the design hierarchy are integrated first. In this example shown in fig. modules 1, 2, 6, 7/8 will be integrated first. Next, modules 1, 3, 4/5 will be integrated. 2. Breadth first Integration: In this type, all modules directly subordinate at each level, moving across the design hierarchy horizontally, are integrated first. In the example shown in figure modules 2 and 3 will be integrated first. Next, modules 6,4 and 5 will be integrated . Modules 7 and 8 will be integrated last. <p>Procedure:</p> <p>The procedure for Top-Down integration process is discussed in the following steps:</p> <ol style="list-style-type: none"> 1. Start with the top or initial module in the software. Substitute the stubs for all the subordinate of the top module. Test the top module. 2. After testing the top module, stubs are replaced one at a time with the actual modules for integration. 3. Perform testing on this recent integrated environment. 4. Regression testing may be conducted to ensure that new errors have not appeared. 5. Repeat steps 2-4 for whole design hierarchy. 	<p>(Description: 2 marks, Labelled Diagram: 2 marks)</p>



(d) What is test plan? Write its two advantage.

4M

Ans: **Test planning** is the first activity of test team. If a tester does not plan for testing then it leads to failure. Test plans are defined in framework created by test strategy and established by test policy Like any project, the testing also should be driven by a plan. The test plan acts as the anchor for the execution, tracking and reporting of the entire testing project. The test plan acts as the anchor for the execution, tracking and reporting of the entire testing project and covers.

(Definition/
Concept: 2
marks,
Advantage:
1 mark
each)

1. Preparing test plan:

- i. What needs to be tested – the scope of testing, including clear identification of what will be the tested & what will not be tested.
- ii. How the testing is going to be performed – breaking down the testing into small and manageable tasks and identifying the strategies to be used for carrying out the tasks.
- iii. What resources are needed for testing- computer as well as human resources.
- iv. The time lines by which the testing activities will be performed.
- v. Risks that may be faced in all of the above, with appropriate mitigation and contingency plans.

2. Scope management: It entails:

- i. Understanding what constitutes a release of product.
- ii. Breaking down the release into features.
- iii. Prioritizing the feature of testing.
- iv. Deciding which features will be tested & which will not be



3. Deciding Test approach/ strategy: This includes identifying.

- i. What type of testing would use for testing functionality?
- ii. What are the configurations for testing features?
- iii. What integration testing would you do to ensure these features work together?
- iv. What “non-functional” tests would you need to do?

4. Setting up criteria for testing: Some of the typical suspension criteria include:

- i. Encountering more than a certain numbers of defects, causing frequent stoppage of testing activity.
- ii. Hitting show stoppers that prevent further progress of testing.

5. Identifying responsibilities, staffing & Training needs:

The next aspect of planning is who part of it. Identifying responsibilities, staffing & training needs addresses this aspect.

6. Identifying Resource Requirement:

As a part of planning for a testing project, the project manager should provide estimate for the various h/w & s/w resources required.

7. Identifying Test Deliverables:

It includes : test plan itself, test case design specification, test cases , test logs & test summary report

8. Testing task:

Size and Effort estimation: This gives estimation in terms of size, effort & schedule of testing project.

Advantages of test planning:

- Work involved in test planning and setup pays in the long term. It gives insight testing activity completely. One knows scope and deliverables of test plan execution.
- Test plan describes the way in which testing team will show whether software work correctly as per requirements and the acceptance criterial as defined b customer or development team with customer. It defines various objectives for testing to measure its performance and coverage offered.
- Test plan addresses various levels of testing such as unit testing module testing, System testing, integration testing, black box testing as well as white box testing. Some time there may be single master test plan with several child test plan at each level for a number of a small plans or one monolithic test plan covering every aspect of testing.
- Test plan explain who does testing. Why test are performed how test are conducted and when tests are scheduled (calendar date and milestone). It defines various criteria such as entry criteria, exit criteria, suspension criteria and resumption criteria at various stages of testing.
- Test plan must contain procedures, environment and tools necessary to implement an orderly, controlled process for test execution, defect tracking, coordination of rework and configuration, and change control.

OR

It is strategic document describes how to perform a task in as effective, efficient and optimized manner. It also specifies, the scope and objectives for testing



Primary tasks in testing are :(Explanation of followings related to):

1. Scope of testing
2. Set objectives of test planning
3. Methodology for testing
4. Requirement,
5. Entry and exit criteria
6. Develop Test matrix
7. Test estimation and administrative component
8. Write and execute test cases.
9. Deciding Criteria for test-to-pass, test-to-fail,
10. Schedule of testing
11. The main purpose of test planning is to show whether software is correct as per requirement.

(B) Attempt any ONE of the following :

06 Marks

(a) Prepare six test cases for home page of marketing site www.flipkart.com.

6M

Ans: Test Cases for Home page of marketing site www.flipkart.com.

Sr. No.	Test Case -ID	Test case Objective	Prerequisite	Steps	Input data	Expected Result	Actual Result	Remarks/ Status
1	TC-1	To check internet	Whether internet is available or not?	Test internet connection	www.flipkart.com	Site home page display should display on screen	Home page displayed	Test to fail
2	TC-2	User name	correct and valid user name should be registered on flipkart	Type correct and valid user name	User name given by flipkart	Should enter valid the user name	Goes to next page if user id and password verified and valid	Test to pass

(Proper test cases in prescribed format or similar Contents: 1 mark Each, Minimum six test cases for given web site or any other relevant test cases)



MODEL ANSWER

SUMMER- 17 EXAMINATION

Subject Title: **SOFTWARE TESTING**

Subject Code:

17624

								ated	
3	TC-3	Password	correct and valid password should be validate and verified on flipkart site	Type correct and valid password	Password selected by user and validated by flipkart	Should enter valid password	Goes to next page if user id and password verified and validated	Test to pass	
4	TC-4	To check whether site home page opens or not	To display flipkart home page website on screen	Type site proper addresses as www.flipkart.com	www.flipkart.com	Site home page display should display on screen	Home page displayed	Test to fail	
5	TC-5	To search the product	Search the product	Click on product link	Mouse rollover and click	Shall display availability of product	Show availability of product	Test to pass	
6	TC-6	Product details	Product shall be available on site	Click on product link	Mouse rollover and click	Shall display product details	displays product details of other product	Test to fail	
7	TC-7	Product details	Product shall be available on site	Click on product link	Mouse rollover and click	Shall display product details	displays product details of prod	Test to pass	



MODEL ANSWER
SUMMER- 17 EXAMINATION

Subject Title: **SOFTWARE TESTING**

Subject Code:

17624

							uct	
8	TC-8	Search for offers	Offer should be valid for that day	Click on Offer link	Select Offer icon and click	Shall display offer details	displays offer details of selected product	Test to pass

Note: Similar test cases relevant to filpkart shall be considered as answer or evidence return as test cases

(b) Enlist any six attributes of defect. Describe them with suitable example.

6M

Ans:

- Defect has following attributes:
Any six of the following attributes shall be considered
- 1) **Defect ID:** Identifies defect as there are many defects might identified in system.
 - a. i.e. D1, D2, etc.
 - 2) **Defect Name:** Name of defect which explains the defect in brief.
 - a. It must be short but descriptive. i.e. Login error.
 - 3) **Project Name:** Indicates project name in which defect is found
 - 4) **Module /Sub-module name:** for which the defect is found.
 - 5) **Phase introduced:** Phase of life cycle to which the defect belongs to.
 - 6) **Phase found:** Phase of project when the defect is found is added here. It is used to find defect leakage or stage.
 - 7) **Defect type:** Defines defect type. i.e. security defect, functional defect, GUI defect etc.
 - 8) **Severity:** Declared in test plan, i.e. high medium or low.
 - 9) **Priority:** defines on the basis of how the project decides a schedule to take the defects for fixing.
 - 10) **Summary:** Describes short about the defect.
 - 11) **Description:** Describes it in detail.
 - 12) **Status:** dynamic field, open, assigned, resolved, closed, hold, deferred, or reopened, etc.
 - 13) **Reported by/ Reported on:** Who found defect, and on what date.
 - 14) **Assigned to:** The tester is being assigned to some testing team member.

(List of attributes of defects: 2 marks, Explanation of attribute minimum 4 defects: 1 mark each.)

OR

ID	Unique identifier given to the defect. (Usually Automated)
Project	Project name.
Product	Product name.



MODEL ANSWER
SUMMER- 17 EXAMINATION

Subject Title: SOFTWARE TESTING

Subject Code:

17624

Release Version	Release version of the product. (e.g. 1.2.3)
Module	Specific module of the product where the defect was
Detected Build Version	Build version of the product where the defect was detected
Summary	Summary of the defect. Keep this clear and concise.
Description	Detailed description of the defect. Describe as much as possible but without repeating anything or using complex words. <i>Keep it simple but comprehensive</i>
Steps to Replicate	Step by step description of the way to reproduce the defect. Number the steps.
Actual Result	The actual result you received when you followed the steps.
Expected Results	The expected results.

2. **Attempt any FOUR of the following :** **16 Marks**

(a) **Describe equivalence partitioning with example.** **4M**

Ans: Equivalence partitioning is a software technique that involves identifying a small set of representative input values that produce as much different output condition as possible. This reduces the number of permutation & combination of input, output values used for testing, thereby increasing the coverage and reducing the effort involved in testing. The set of input values that generate one single expected output is called a partition. When the behavior of the software is the same for a set of values, then the set is termed as equivalence class or partition.

Example: An insurance company that has the following premium rates based on the age group. A life insurance company has base premium of Rs. 500 for all ages. Based on the age group, an additional monthly premium has to pay that is as listed in the table below. For example , a person aged 34 has to pay a premium=**Rs. 500 + Rs. 1000=Rs. 1500**

Age group	Extra Premium
Under 35	Rs.1500
35-59	Rs. 2500
60+	Rs. 4000

Based on the equivalence partitioning technique, the equivalence partitions that are based on age are given below:

- Below 35 years of age (valid input)
- Between 35 and 59 years of age (valid input)
- Above 6 years of age (valid input)
- Negative age (invalid input)

(Explanation: 3 marks, Example:1 mark)



	<ul style="list-style-type: none"> • Age as 0(invalid input) • Age as any three-digit number(valid input) 	
(b)	State & explain any four benefits of automation in testing.	4M
Ans:	<p>Benefits of automation testing:</p> <ol style="list-style-type: none"> 1. Speed: Think about how long it would take you to manually try a few thousand test cases for the windows Calculator. You might average a test case every five seconds or so. Automation might be able to run 10, 100 even 1000 times that fast. 2. Efficiency: While you are busy running test cases, you can't be doing anything else. If you have a test tool that reduces the time it takes for you to run your tests, you have more time for test planning and thinking up new tests. 3. Accuracy and Precision: After trying a few hundred cases, your attention may reduce and you will start to make mistakes .A test tool will perform the same test and check the result perfectly, each and every time. 4. Resource Reduction: Sometimes it can be physically impossible to perform a certain test case. The number of people or the amount of equipment required to create the test condition could be prohibitive. A test tool can used to simulate the real world and greatly reduce the physical resources necessary to perform the testing. 5. Simulation and Emulation: Test tools are used to replace hardware or software that would normally interface to your product. This “face” device or application can then be used to drive or respond to your software in ways that you choose-and ways that might otherwise be difficult to achieve. 6. Relentlessness: Test tool and automation never tire or give up. It will continuously test the software. <p style="text-align: center;">OR</p> <p>Benefits of Automation Testing are:</p> <ol style="list-style-type: none"> 1. Save Time /Speed: Due to advanced computing facilities, automation test tools prevail in speed of processing the tests. Automation saves time as software can execute test cases faster than human. 2. Reduces the tester's involvement in executing tests: It relieves the testers to do some other work. 3. Repeatability/Consistency: The same tests can be re-run in exactly the same manner eliminating the risk of human errors such as testers forgetting their exact actions, intentionally omitting steps from the test scripts, missing out steps from the test script, all of which can result in either defects not being identified or the reporting of invalid bugs (which can again, be time consuming for both developers and testers to reproduce) 4. Simulated Testing: Automated tools can create many concurrent virtual users/data and effectively test the project in the test environment before releasing the product. 5. Test case design: Automated tools can be used to design test cases also. Through automation, better coverage can be guaranteed than if done manually. 6. Reusable: The automated tests can be reused on different versions of the 	(State: 1 mark, Explanation : 3 marks)



MODEL ANSWER
SUMMER- 17 EXAMINATION

Subject Title: **SOFTWARE TESTING**

Subject Code: **17624**

	<p>software, even if the interface changes.</p> <p>7. Avoids human mistakes: Manually executing the test cases may incorporate errors. But this can be avoided in automation testing.</p> <p>8. Internal Testing: Testing may require testing for memory leakage or checking the coverage of testing. Automation can done this easily.</p> <p>9. Cost Reduction: If testing time increases cost of the software also increases. Due to testing tools time and therefore cost is reduced.</p>									
(c)	How to perform security testing? State elements of security testing.	4M								
Ans:	<p>Security Testing: Testers must use a risk-based approach, By identifying risks and potential loss associated with those risks in the system and creating tests driven by those risks, the testers can properly focus on areas of code in which an attack is likely to succeed. Therefore risk analysis at the design level can help to identify potential security problems and their impacts. Once identified ranked, software risks can help guide software security. It is a type of non-functional testing. Security testing is basically a type of software testing that's done to check whether the application or the product is secured or not. It checks to see if the application is vulnerable to attacks, if anyone hack the system or login to the application without any authorization. It is a process to determine that an information system protects data and maintains functionality as intended. The security testing is performed to check whether there is any information leakage in the sense by encrypting the application or using wide range of software's and hardware's and firewall etc.</p> <p>Software security is about making software behave in the presence of a malicious attack.</p> <p>The six basic security concepts / elements that need to be covered by security testing are:</p> <ul style="list-style-type: none"> • confidentiality, • integrity, • authentication, • availability, • authorization and • Non-repudiation. 	(Perform security Testing: 2 marks, List of elements of Security Testing: 2 marks)								
(d)	What is the difference between static & dynamic testing tool?	4M								
Ans:	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 50%; text-align: center;">Static testing tool</th> <th style="width: 50%; text-align: center;">Dynamic testing tool</th> </tr> </thead> <tbody> <tr> <td>These tools are used by developers as part of the development and component testing process</td> <td>These tools require the code to be in a "running state"</td> </tr> <tr> <td>code is not executed or run but tool itself is executed</td> <td>They analyse rather than testing</td> </tr> <tr> <td>It is extension of compiler technology</td> <td>They also help to understand background processes</td> </tr> </tbody> </table>	Static testing tool	Dynamic testing tool	These tools are used by developers as part of the development and component testing process	These tools require the code to be in a "running state"	code is not executed or run but tool itself is executed	They analyse rather than testing	It is extension of compiler technology	They also help to understand background processes	(Four points of difference: 1 mark each)
Static testing tool	Dynamic testing tool									
These tools are used by developers as part of the development and component testing process	These tools require the code to be in a "running state"									
code is not executed or run but tool itself is executed	They analyse rather than testing									
It is extension of compiler technology	They also help to understand background processes									



MODEL ANSWER

SUMMER- 17 EXAMINATION

Subject Title: SOFTWARE TESTING

Subject Code:

17624

		<p>It also perform static analysis of requirement or analysis of website</p>	<p>These tool used by developed in component integration testing,, middle ware , testing robustness and security.</p>		
		<p>Helps to understand the structure of the code and can also be useful to enforce coding standards.</p>	<p>Also performs web site testing to check whether each link does actually link to something else, it can find dead links .</p>		
		<p>Features /characteristics of static testing tools are:</p> <ul style="list-style-type: none"> • Checks cyclomatic complexity • Enforces coding standards • Analyse structures and dependencies • Helpful in understanding coding • Identify defects in code. 	<p>Features/characteristics of static testing tools are:</p> <ul style="list-style-type: none"> • Detect memory leak • Identify pointer arithmetic errors , null pointer • Identify time dependence. 		
		<p>Examples. Flow analyzer, path tests, coverage analyzers, Interface analyzers</p>	<p>Examples. Test driver, Test beds, Emulators, Mutation analyzers</p>		
	(e)	Describe positive testing & negative testing. Also write test cases for them.			4M
Ans:	<p>Software testing is process of Verification and Validation to check whether software application under test is working as expected. To test the application we need to give some input and check if getting result as per mentioned in the requirements or not. This testing activity is carried out to find the defects in the code & improve the quality of software application. Testing of application can be carried out in two different ways, Positive testing and Negative testing.</p> <p>Positive Testing: Positive Testing is testing process where the system validated against the valid input data. In this testing tester always check for only valid set of values and check if a application behaves as expected with its expected inputs. The main intention of this testing is to check whether software application not showing error when not supposed to & showing error when supposed to. Such testing is to be carried out keeping positive point of view & only execute the positive scenario. Positive Testing always tries to prove that a given product and project always meets the requirements and specifications. Under Positive testing is test the normal day to day life scenarios and check the expected behavior of application.</p> <p>Negative Testing: Negative Testing is testing process where the system validated against the invalid input data. A negative test checks if a application behaves as expected with its negative inputs. The main intention of this testing is to check whether software application not showing error when supposed to & showing error when not supposed to. Such testing is to be carried out keeping negative point of view & only execute the test cases for only invalid set of input data. Negative testing is a testing process to identify the inputs where</p>				<p>(Description of positive and negative testing: 2 marks, Test cases/example-2marks or any other relevant example)</p>



MODEL ANSWER

SUMMER- 17 EXAMINATION

Subject Title: **SOFTWARE TESTING**

Subject Code: **17624**

system is not designed or un-handled inputs by providing different invalid. The main reason behind Negative testing is to check the stability of the software application against the influences of different variety of incorrect validation data set.

Test case for text box which accepts input values from user:

Sr. No.	Test Case-ID	Test case Objective	Prerequisite	Steps	Input data	Expected Result	Actual Result	Remark
1	TC-1.1	Positive Testing	Text box should be present and it should accept numeric values only	Enter numbers/digits in text box i.e. 123 entered	0, 1 to 9 or combination of these numbers	Able to take input as numbers	Text box accepts numeric values	Test to pass
2	TC2.1	Negative Testing	Text box should be present and it should accept numeric values only	Enter numbers/digits in text box i.e. abc entered	0, 1 to 9 or combination of these numbers	Able to take input as only numeric values	Text box accepts all characters..	Test to fail

Some more scenarios for testing:

Positive Test Scenarios:

- Password textbox should accept 6 characters
- Password textbox should up to 20 characters
- Password textbox should accept any value in between 6-20 char's length.
- Password textbox should accept all numeric & alphabets values.

Negative Test scenarios:

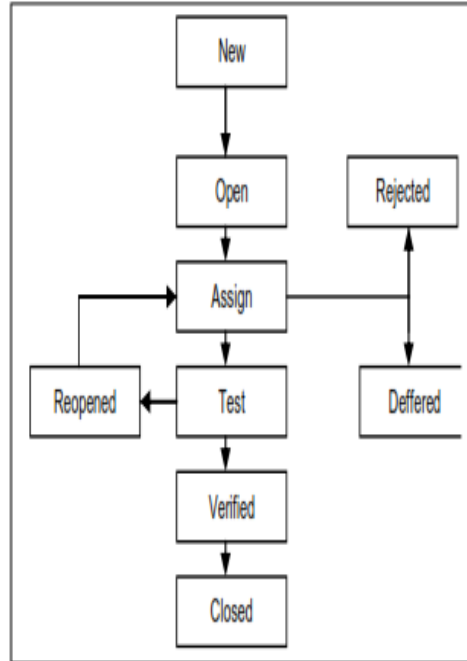
- Password textbox should not accept less than 6 characters
- Password textbox should not exceeds more than 20 characters
- Password textbox should not accept special characters.



(f) Describe defect life cycle with neat diagram.

4M

Ans:



OR

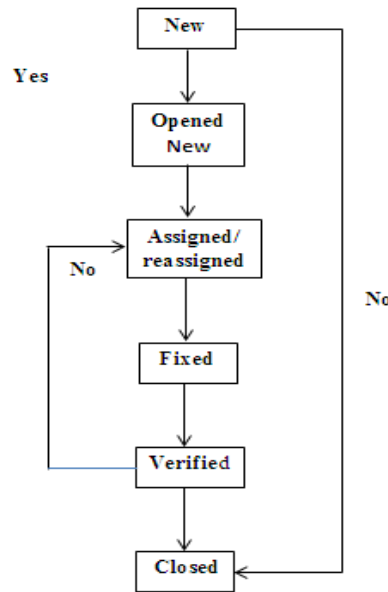


Fig: Defect life cycle

The different states of bug life cycle are as shown in the above diagram:

- **New:** When the bug is posted for the first time, its state will be “NEW”. This means that the bug is not yet approved.

(Description of defect life cycle Stages: 2 marks, Proper Labelled Diagram: 2 marks)

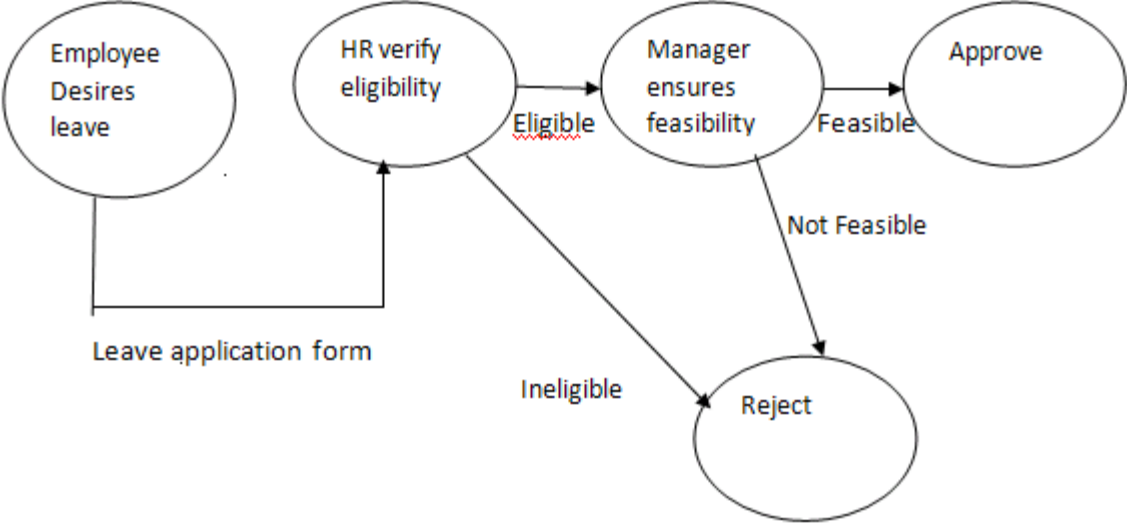


- **Open:** After a tester has posted a bug, the lead of the tester approves that the bug is genuine and he changes the state as “OPEN”.
- **Assign:** Once the lead changes the state as “OPEN”, he assigns the bug to corresponding developer or developer team. The state of the bug now is changed to “ASSIGN”.
- **Test/Retest:** Once the developer fixes the bug, he has to assign the bug to the testing team for next round of testing. Before he releases the software with bug fixed, he changes the state of bug to “TEST”. It specifies that the bug has been fixed and is released to testing team.// At this stage the tester do the retesting of the changed code which developer has given to him to check whether the defect got fixed or not.
- **Deferred:** The bug, changed to deferred state means the bug is expected to be fixed in next releases. The reasons for changing the bug to this state have many factors. Some of them are priority of the bug may be low, lack of time for the release or the bug may not have major effect on the software.
- **Rejected:** If the developer feels that the bug is not genuine, he rejects the bug. Then the state of the bug is changed to “REJECTED”.
- **Verified:** Once the bug is fixed and the status is changed to “TEST”, the tester tests the bug. If the bug is not present in the software, he approves that the bug is fixed and changes the status to “VERIFIED”.
- **Reopened:** If the bug still exists even after the bug is fixed by the developer, the tester changes the status to “REOPENED”. The bug traverses the life cycle once again.
- **Closed:** Once the bug is fixed, it is tested by the tester. If the tester feels that the bug no longer exists in the software, he changes the status of the bug to “CLOSED”. This state means that the bug is fixed, tested and approved.
- **Fixed:** When developer makes necessary code changes and verifies the changes then he/she can make bug status as „Fixed“ and the bug is passed to testing team.
- **Pending retest:** After fixing the defect the developer has given that particular code for retesting to the tester. Here the testing is pending on the testers end. Hence its status is pending retest.

Optional :

- **Duplicate:** If the bug is repeated twice or the two bugs mention the same concept of the bug, then one bug status is changed to “duplicate“.
- **Not a bug:** The state given as “Not a bug” if there is no change in the functionality of the application. For an example: If customer asks for some change in the look and field of the application like change of color of some text then it is not a bug but just some change in the looks of the application.



3.	Attempt any FOUR of the following :	16 Marks
	(a) Illustrate process of graph-based testing with suitable example.	4M
Ans:	<p>i. Black-box methods based on the nature of the relationships (links) among the program objects (nodes), test cases are designed to traverse the entire graph</p> <p>ii. Transaction flow testing – nodes represent steps in some transaction and links represent logical connections between steps that need to be validated</p> <p>iii. Finite state modeling – nodes represent user observable states of the software and links represent transitions between states</p> <p>iv. Data flow modeling – nodes are data objects and links are transformations from one data object to another</p> <p>v. Timing modeling – nodes are program objects and links are sequential connections between these objects, link weights are required execution times.</p> <p>Steps in graph testing:</p> <p>i. Build a graph model.</p> <p>ii. Identify the test requirements.</p> <p>iii. Select test paths to cover those requirements.</p> <p>Derive test data so that those test paths can be executed.</p> 	(Explanation: 2 marks, Example: 2 marks, Any other relevant example shall be considered)
	(b) Describe any two special tests in testing process.	4M
Ans:	<p>1. Smoke Testing: is a testing technique that is inspired from hardware testing, which checks for the smoke from the hardware components once the hardware's power is switched on. ii. In Software testing context, smoke testing refers to testing the basic functionality of the build. iii. If the Test fails, build is declared as unstable and it is NOT tested anymore until the smoke test of the build passes.</p> <p>Smoke Testing - Features:</p> <p>i. Identifying the business critical functionalities that a product must satisfy.</p> <p>ii. Designing and executing the basic functionalities of the application.</p>	(Any two special tests: 2 marks each)



iii. Ensuring that the smoke test passes each and every build in order to proceed with the testing.

iv. Smoke Tests enables uncovering obvious errors which saves time and effort of test team.

v. Smoke Tests can be manual or automated.

2. Sanity testing,: A software testing technique performed by the test team for some basic tests. The aim of basic test is to be conducted whenever a new build is received for testing. The terminologies such as Smoke Test or Build Verification Test or Basic Acceptance Test or Sanity Test are interchangeably used, however, each one of them is used under a slightly different scenario. **ii.** Sanity test is usually unscripted, helps to identify the dependent missing functionalities. It is used to determine if the section of the application is still working after a minor change. **iii.** Sanity testing can be narrow and deep. Sanity test is a narrow regression test that focuses on one or a few areas of functionality.

3. Regression Testing: Regression testing a black box testing technique that consists of re-executing those tests that are impacted by the code changes. **ii.** These tests should be executed as often as possible throughout the software development life cycle. Types of Regression Tests: **i.** Final Regression Tests: - A "final regression testing" is performed to validate the build that hasn't changed for a period of time. This build is deployed or shipped to customers. **ii.** Regression Tests: - A normal regression testing is performed to verify if the build has NOT broken any other parts of the application by the recent code changes for defect fixing or for enhancement.

4. Usability Testing. **i.** Usability testing, a non-functional testing technique that is a measure of how easily the system can be used by end users. **ii.** It is difficult to evaluate and measure but can be evaluated based on the below parameters: **iii.** Level of Skill required to learn/use the software. It should maintain the balance for both novice and expert user. **iv.** Time required to get used to in using the software. **v.** The measure of increase in user productivity if any. **vi.** Assessment of a user's attitude towards using the software.

5. GUI Testing. **i.** GUI testing is a testing technique in which the application's user interface is tested whether the application performs as expected with respect to user interface behavior. **ii.** GUI Testing includes the application behavior towards keyboard and mouse movements and how different GUI objects such as toolbars, buttons, menu bars, dialog boxes, edit fields, lists, behavior to the user input.

GUI Testing Guidelines: **i.** Check Screen Validations **ii.** Verify All Navigations **iii.** Check usability Conditions **iv.** Verify Data Integrity **v.** Verify the object states **vi.** Verify the date Field and Numeric Field Formats.

6. Object Oriented Application Testing. **i.** The Full-Lifecycle Object-Oriented Testing



(FLOOT) methodology is a collection of testing techniques to verify and validate object-oriented software. ii. The FLOOT lifecycle is depicted in Figure 9, indicating a wide variety of techniques (described in Table 9 are available to you throughout all aspects of software development. iii. The list of techniques is not meant to be complete: instead the goal is to make it explicit that you have a wide range of options available to you. iv. It is important to understand that although the FLOOT method is presented as a collection of serial phases it does not need to be so: the techniques of FLOOT can be applied with evolutionary/agile processes as well. v. The reason why I present the FLOOT in a "traditional" manner is to make it explicit that you can in fact test throughout all aspects of software development, not just during coding.

7. Client Server Testing. i. This type of testing usually done for 2 tier applications (usually developed for LAN) Here we will be having front-end and backend. ii. The application launched on front-end will be having forms and reports which will be monitoring and manipulating data.E.g: applications developed in VB, VC++, Core Java, C, C++, D2K, PowerBuilder etc., iii. The backend for these applications would be MS Access, SQL Server, Oracle, Sybase, Mysql, Quadbase. iv. The tests performed on these types of applications would be– User interface testing Manual support testing– Functionality testing– Compatibility testing & configuration testing – Intersystem testing.

8. Web Based Testing. i. Web application testing, a software testing technique exclusively adopted to test the applications that are hosted on web in which the application interfaces and other functionalities are tested.

Web Application Testing Techniques:

1. Functionality Testing
2. Usability testing
3. Interface testing
4. Compatibility testing
5. Performance testing
6. Security testing

(c) How to identify resource requirement of test plan?

4M

Ans: Resource requirement is a **detailed summary** of all types of resources required to complete project task. Resource could be human, equipment and materials needed to complete a project. The resource requirement and planning is important factor of the test planning because helps in **determining** the **number** of resources (employee, equipment...) to be used for the project. Therefore, the Test Manager can make the correct schedule & estimation for the project.

(Any 4 resources: 4 marks, Any other relevant answer)

Some of the following factors need to be considered:

- Machine configuration (RAM,processor,disk)needed to run the product under test.

**MODEL ANSWER****SUMMER- 17 EXAMINATION**Subject Title: **SOFTWARE TESTING**

Subject Code:

17624

- Overheads required by test automation tools, if any
- Supporting tools such as compilers, test data generators, configuration management tools.
- The different configurations of the supporting software(e.g. OS)that must be present

No.	Member	Tasks
1.	Test Manager	Manage the whole project Define project directions Acquire appropriate resources
2.	Tester	Identifying and describing appropriate test techniques/tools/automation architecture Verify and assess the Test Approach Execute the tests, Log results, Report the defects. Tester could be in-sourced or out-sourced member base on the project budget For the task which required low skill, I recommend you choose outsourced members to save project cost.
3.	Developer in Test	Implement the test cases, test program, test suite etc.
4.	Test Administrator	Builds up and ensures test environment and assets are managed and maintained Support Tester to use the test environment for test execution



5.

SQA members

Take in charge of quality assurance

Check to confirm whether the testing process is meeting specified requirements

- Special requirements for running machine-intensive tests such as load tests and performance tests.
- Appropriate number of licenses of all the software

OR

Human Resource:

The following table represents various members in your project team

System Resource:

For testing, a web application, you should plan the resources as following tables:

No.	Resources	Descriptions
1.	Server	Install the web application under test This includes a separate web server, database server, and application server if applicable
2.	Test tool	The testing tool is to automate the testing, simulate the user operation, generate the test results There are tons of test tools you can use for this project such as Selenium QTP...etc.
3.	Network	You need a Network include LAN and Internet to simulate the real business and user environment
4.	Computer	The PC which users often use to connect the web server



	(d)	Illustrate process of bi-directional integration testing. State its two advantages & disadvantages.	4M
Ans:		<ol style="list-style-type: none">1. Bi-directional Integration, is a kind of integration testing process that combines top-down and bottom-up testing.2. With an experience in delivering Bi-directional testing projects custom software development services provide the best quality of the deliverables right from the development of software process.3. Bi-directional Integration testing is a vertical incremental testing strategy that tests the bottom layers and top layers and tests the integrated system in the computer software development process.4. Using stubs, it tests the user interface in isolation as well as tests the very lowest level functions using drivers.5. Bi-directional Integration testing combines bottom-up and top-down testing.6. Bottom-up testing is a process where lower level modules are integrated and then tested.7. This process is repeated until the component of the top of the hierarchy is analyzed. It helps custom software development services find bugs easily without any problems.8. Top down testing is a process where the top integrated modules are tested and the procedure is continued till the end of the related module.9. Top down testing helps developers find the missing branch link easily. <p style="text-align: center;">OR</p> <p>Process of Bidirectional testing:</p> <ol style="list-style-type: none">1. Bottom up testing starts from middle layer and goes upward to the top layer. For a very big system, bottom up approach starts at a subsystem level and goes upwards.2. Top down testing starts from the middle layer and goes downward. For a very big system, top down approach, starts at subsystem level and goes downwards.3. Big band approach is followed for middle layer. From this layer, bottom up approach goes upwards and top down approach goes downwards. <p>Advantages:</p> <ol style="list-style-type: none">1. This approach is useful is useful for very large projects having several projects. When development follows a spiral model and module itself is as large as a system.2. Both top down and bottom up approach starts at the start of the schedule.3. It needs more resources and big teams for performing both, methods of testing at a time or one after the other. <p>Disadvantages:</p> <ol style="list-style-type: none">1. It represents very high cost of testing as lot of testing is done.2. It cannot be used for smaller systems with huge interdependence between different modules.3. Different skill tests are required for testers at different level as modules are separate systems handling separate domains.	(Explanati on: 2 marks, Advantage s: 1 mark, Disadvant ages: 1 mark)



MODEL ANSWER
SUMMER- 17 EXAMINATION

Subject Title: SOFTWARE TESTING

Subject Code: 17624

	(e)	Differentiate between manual testing & automation testing.	4M														
	Ans:	<table border="1"> <thead> <tr> <th data-bbox="212 331 667 390">Automated Testing</th> <th data-bbox="667 331 1430 390">Manual Testing</th> </tr> </thead> <tbody> <tr> <td data-bbox="212 390 667 527">• If you have to run a set of tests repeatedly automation is a huge gain</td> <td data-bbox="667 390 1430 527">• If Test Cases have to be run a small number of times it's more likely to perform manual testing</td> </tr> <tr> <td data-bbox="212 527 667 646">• Helps performing "compatibility testing" - testing the software on different configurations</td> <td data-bbox="667 527 1430 646">• It allows the tester to perform more ad-hoc (random testing)</td> </tr> <tr> <td data-bbox="212 646 667 766">• It gives you the ability to run automation scenarios to perform regressions in a shorter time</td> <td data-bbox="667 646 1430 766">• Short term costs are reduced</td> </tr> <tr> <td data-bbox="212 766 667 886">• It gives you the ability to run regressions on a code that is continuously changing</td> <td data-bbox="667 766 1430 886">• The more time tester spends testing a module the grater the odds to find real user bugs</td> </tr> <tr> <td data-bbox="212 886 667 1005">• It's more expensive to automate. Initial investments are bigger than manual testing</td> <td data-bbox="667 886 1430 1005">• Manual tests can be very time consuming</td> </tr> <tr> <td data-bbox="212 1005 667 1129">• You cannot automate everything, some tests still have to be done manually</td> <td data-bbox="667 1005 1430 1129">• For every release you must rerun the same set of tests which can be tiresome</td> </tr> </tbody> </table>	Automated Testing	Manual Testing	• If you have to run a set of tests repeatedly automation is a huge gain	• If Test Cases have to be run a small number of times it's more likely to perform manual testing	• Helps performing "compatibility testing" - testing the software on different configurations	• It allows the tester to perform more ad-hoc (random testing)	• It gives you the ability to run automation scenarios to perform regressions in a shorter time	• Short term costs are reduced	• It gives you the ability to run regressions on a code that is continuously changing	• The more time tester spends testing a module the grater the odds to find real user bugs	• It's more expensive to automate. Initial investments are bigger than manual testing	• Manual tests can be very time consuming	• You cannot automate everything, some tests still have to be done manually	• For every release you must rerun the same set of tests which can be tiresome	(Any 4 Points: 1 mark Each)
Automated Testing	Manual Testing																
• If you have to run a set of tests repeatedly automation is a huge gain	• If Test Cases have to be run a small number of times it's more likely to perform manual testing																
• Helps performing "compatibility testing" - testing the software on different configurations	• It allows the tester to perform more ad-hoc (random testing)																
• It gives you the ability to run automation scenarios to perform regressions in a shorter time	• Short term costs are reduced																
• It gives you the ability to run regressions on a code that is continuously changing	• The more time tester spends testing a module the grater the odds to find real user bugs																
• It's more expensive to automate. Initial investments are bigger than manual testing	• Manual tests can be very time consuming																
• You cannot automate everything, some tests still have to be done manually	• For every release you must rerun the same set of tests which can be tiresome																
4.	(A)	Attempt any THREE of the following :	12 Marks														
	(a)	Define software metrics. Describe product Vs process & objective Vs subjective metrics.	4M														
	Ans:	<p>Metrics are necessary to provide measurements of such qualities. Metrics can also be used to gauge the size and complexity of software and hence are employed in project management and cost estimation.</p> <p>Process quality: Activities related to the production of software, tasks or milestones.</p> <ol style="list-style-type: none"> 1. Process metrics are collected across all projects and over long periods of time. 2. They are used for making strategic decisions. 3. The intent is to provide a set of process indicators that lead to long-term software process improvement. 4. The only way to know how/where to improve any process is to: <ul style="list-style-type: none"> • Measure specific attributes of the process. • Develop a set of meaningful metrics based on these attributes. • Use the metrics to provide indicators that will lead to a strategy for improvement. <p>Product quality: Explicit result of the software development activity, deliverables, products.</p> <ol style="list-style-type: none"> 1. Product metrics help software engineers to better understand the attributes of models and assess the quality of the software. 2. They help software engineers to gain insight into the design and construction of the 	(Definition :1 mark, Product Vs Process:1 ½ marks, Objective Vs Subjective :1 ½ marks)														



	<p>software.</p> <ol style="list-style-type: none"> 3. Focus on specific attributes of software engineering work products resulting from analysis, design, coding, and testing. 4. Provide a systematic way to assess quality based on a set of clearly defined rules. 5. Provide an “on-the-spot” rather than “after-the-fact” insight into the software development. <p>Objective Metrics:</p> <ol style="list-style-type: none"> 1. They are non-negotiable – that is the way they are defined doesn’t change with respect to the niche or the type of endeavor they are being applied to. 2. Actual cost or AC is always the total cost actually incurred in accomplishing a certain activity or a sequence of activities. <p>Subjective Metrics:</p> <ol style="list-style-type: none"> 1. These metrics are a relatively new precept and are more flexible than the rigid framework of the objective metrics. Subjective metrics do deal with performance but the approach is more tailored. For some enterprises the niche in which they function forces project management to change in order to adapt to the demands of the workplace. 	
	<p>(b) How to prepare test plan?</p>	<p>4M</p>
<p>Ans:</p>	<p>Steps to prepare test plan:</p> <p>Like any project, the testing also should be driven by a plan. The test plan acts as the anchor for the execution, tracking and reporting of the entire testing project. Activities of test plan:</p> <ol style="list-style-type: none"> 1. Scope Management: Deciding what features to be tested and not to be tested. 2. Deciding Test approach /strategy: Which type of testing shall be done like configuration, integration, localization etc. 3. Setting up criteria for testing: There must be clear entry and exit criteria for different phases of testing. The test strategies for the various features and combinations determined how these features and combinations would be tested. 4. Identifying responsibilities, staffing and training needs. 5. Identifying resource requirements 6. Identifying test deliverables. 7. Testing tasks: size and effort estimation. <p style="text-align: center;">OR</p> <ol style="list-style-type: none"> 1. Test Plan is a document that is the point of reference based on which testing is carried out within the QA team. 2. Test plan is not static and is updated on an on demand basis. 3. The more detailed and comprehensive the Test plan, the more successful the testing activity. 4. The test plan keeps track of possible tests that will be run on the system after coding. 5. The test plan is a document that develops as the project is being developed. 	<p>(Steps :4 marks or any other relevant answer shall be considered)</p>



6. Record tests as they come up
7. Test error prone parts of software development.
8. The initial test plan is abstract and the final test plan is concrete.
9. The initial test plan contains high level ideas about testing the system without getting into the details of exact test cases.
10. The most important test cases come from the requirements of the system.
11. When the system is in the design stage, the initial tests can be refined a little.
12. During the detailed design or coding phase, exact test cases start to materialize.
13. After coding, the test points are all identified and the entire test plan is exercised on the software.

(c) Describe techniques for finding defects.

4M

Ans: **Static Techniques:** Static techniques of quality control define checking the software product and related artifacts without executing them. It is also termed desk checking/verification /white box testing'. It may include reviews, walkthroughs, inspection, and audits Here; the work product is reviewed by the reviewer with the help of a checklist, standards, any other artifact, knowledge and experience, in order to locate the defect with respect to the established criteria. Static technique is so named because it involves no execution of code, product, documentation, etc. This technique helps in establishing conformance to requirements view.

(Any 2 technique s: 2 mark Each)

Dynamic Testing: Dynamic testing is a validation technique which includes dummy or actual execution of work products to evaluate it with expected behavior. It includes black box testing methodology such as system testing and unit testing. The testing methods evaluate the product with respect to requirements defined, designs created and mark it as pass or fail'. This technique establishes fitness for use'view.

Operational techniques: Operational techniques typically include auditing work products and projects to understand whether the processes defined for development /testing are being followed correctly o not, and also whether they are effective or not. It also includes revisiting the defects before and after fixing and analysis. Operational technique may include smoke testing and sanity testing of a work product.

OR

a)Quick Attacks:

i. Strengths:

- The quick-attacks technique allows you to perform a cursory analysis of a system in a very compressed timeframe.
- Even without a specification, you know a little bit about the software, so the time spent is also time invested in developing expertise.
- The skill is relatively easy to learn, and once you've attained some mastery your quick-attack session will probably produce a few bugs.



- Finally, quick attacks are *quick*.
- They can help you to make a rapid assessment. You may not know the requirements, but if your attacks yielded a lot of bugs, the programmers probably aren't thinking about exceptional conditions, and it's also likely that they made mistakes in the main functionality.
- If your attacks don't yield any defects, you may have some confidence in the general, happy-path functionality.

ii. Weaknesses:

- Quick attacks are often criticized for finding "bugs that don't matter"—especially for internal applications.
- While easy mastery of this skill is a strength, it creates the risk that quick attacks are "all there is" to testing; thus, anyone who takes a two-day course can do the work.

a) Equivalence and Boundary Conditions:**i. Strengths:**

- Boundaries and equivalence classes give us a technique to reduce an infinite test set into something manageable.
- They also provide a mechanism for us to show that the requirements are "covered".

ii. Weaknesses:

- The "classes" in the table in Figure 1 are correct only in the mind of the person who chose them.
- We have no idea whether other, "hidden" classes exist—for example, if a numeric number that represents time is compared to another time as a set of characters, or a "string," it will work just fine for most numbers.

b) Common Failure Modes:**i. Strengths:**

- The heart of this method is to figure out what failures are common for the platform, the project, or the team; then try that test again on this build.
- If your team is new, or you haven't previously tracked bugs, you can still write down defects that "feel" recurring as they occur—and start checking for them.

ii. Weaknesses:

- In addition to losing its potency over time, this technique also entirely fails to find "black swans"—defects that exist outside the team's recent experience.
- The more your team stretches itself (using a new database, new programming language, new team members, etc.), the riskier the project will be—and, at the same time, the less valuable this technique will be.

c) State-Transition Diagrams:

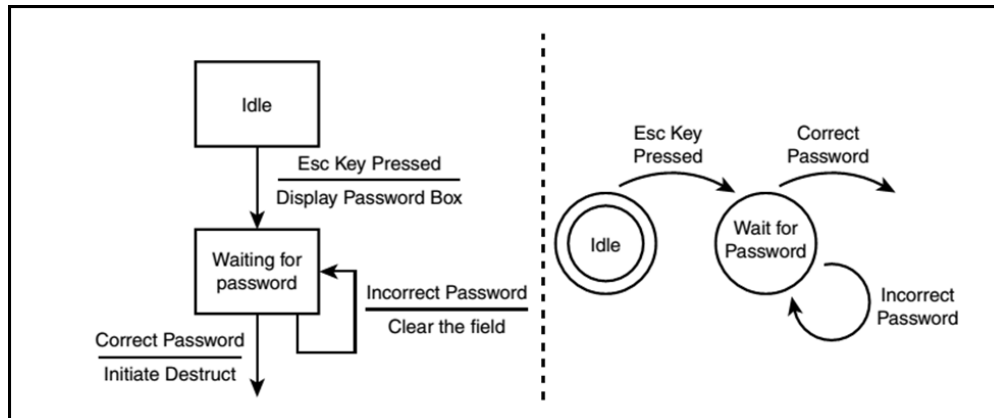


Figure 4: State Transition Map

i. Strengths:

- Mapping out the application provides a list of immediate, powerful test ideas.
- Model can be improved by collaborating with the whole team to find "hidden" states—transitions that might be known only by the original programmer or specification author.
- Once you have the map, you can have other people draw their own diagrams, and then compare theirs to yours.
- The differences in those maps can indicate gaps in the requirements, defects in the software, or at least different expectations among team members.

ii. Weaknesses:

- The map you draw doesn't actually reflect how the software will operate; in other words, "the map is not the territory."
- Drawing a diagram won't find these differences, and it might even give the team the illusion of certainty.
- Like just about every other technique on this list, a state-transition diagram can be helpful, but it's not sufficient by itself to test an entire application.

d) Use Cases and Soap Opera Tests:

Use cases and scenarios focus on software in its role to enable a human being to do something.

i. Strengths:

- Use cases and scenarios tend to resonate with business customers, and if done as part of the requirement process, they sort of magically generate test cases from the requirements.



- They make sense and can provide a straightforward set of confirmatory tests. Soap opera tests offer more power, and they can combine many test types into one execution.

Weaknesses:

- Soap opera tests have the opposite problem; they're so complex that if something goes wrong, it may take a fair bit of troubleshooting to find exactly where the error came from!

- e) **Code-Based Coverage Models:** Imagine that you have a black-box recorder that writes down every single line of code as it executes.

i. Strengths:

- Programmers love code coverage. It allows them to attach a number—an actual, hard, real number, such as **75%**—to the performance of their unit tests, and they can challenge themselves to improve the score.
- Meanwhile, looking at the code that *isn't* covered also can yield opportunities for improvement and bugs!

ii. Weaknesses:

- Customer-level coverage tools are expensive, programmer-level tools that tend to assume the team is doing automated unit testing and has a continuous-integration server and a fair bit of discipline.
- After installing the tool, most people tend to focus on statement coverage—the least powerful of the measures.
- Even decision coverage doesn't deal with situations where the decision contains defects, or when there are other, hidden equivalence classes; say, in the third-party library that isn't measured in the same way as your compiled source code is.
- Having code-coverage numbers can be helpful, but using them as a form of process control can actually encourage wrong behaviours. In my experience, it's often best to leave these measures to the programmers, to measure optionally for personal improvement (and to find dead spots), not as a proxy for actual quality.

f) Regression and High-Volume Test Techniques:

- People spend a lot of money on regression testing, taking the old test ideas described above and rerunning them over and over.
- This is generally done with either expensive users or *very* expensive programmers spending a lot of time writing and later maintaining those automated tests.

i. Strengths:



- For the right kind of problem, say an IT shop processing files through a database, this kind of technique can be extremely powerful.
 - Likewise, if the software deliverable is a report written in SQL, you can hand the problem to other people in plain English, have them write their own SQL statements, and compare the results.
 - Unlike state-transition diagrams, this method shines at finding the hidden state in devices. For a pacemaker or a missile-launch device, finding those issues can be pretty important.
- ii. Weaknesses:**
- Building a record/playback/capture rig for a GUI can be extremely expensive, and it might be difficult to tell whether the application hasn't broken, but has changed in a minor way.
 - For the most part, these techniques seem to have found a niche in IT/database work, at large companies like Microsoft and AT&T, which can have programming testers doing this work in addition to traditional testing, or finding large errors such as crashes without having to understand the details of the business logic.
 - While some software projects seem ready-made for this approach, others...aren't.
 - You could waste a fair bit of money and time trying to figure out where your project falls.

(d) Describe test case specification of test process.

4M

Ans:

Test case specification:

1. The purpose of the test.
2. Items being tested, along with their version/release numbers as appropriate.
3. Environment that need to be set up for running the test case.
4. Input data to be used for the test case.
5. Steps to be followed to execute the test.
6. The expected result that are considered to be "correct result"
7. A steps to compare the actual results produced with the expected results.
8. Any relationship between this test and other tests.

OR

Testing is a process rather than a single activity.

- i. This process starts from test planning then designing test cases, preparing for execution and evaluating status till the test closure.
- ii. Divide the activities within the fundamental test process into the following basic steps:

1. Planning and Control

Test planning has following major tasks:

**(Explanati
on: 4
marks)**



- i. To determine the scope and risks and identify the objectives of testing.
- ii. To determine the test approach.
- iii. To implement the test policy and/or the test strategy. (Test strategy is an outline that describes the testing portion of the software development cycle. It is created to inform PM, testers and developers about some key issues of the testing process. This includes the testing objectives, method of testing, total time and resources required for the project and the testing environments.)
- iv. To determine the required test resources like people, test environments, PCs, etc.
- v. To schedule test analysis and design tasks, test implementation, execution and evaluation.
- vi. To determine the Exit criteria we need to set criteria such as Coverage criteria. (Coverage criteria are the percentage of statements in the software that must be executed during testing. This will help us track whether we are completing test activities correctly. They will show us which tasks and checks we must complete for a particular level of testing before we can say that testing is finished.)

Test control has the following major tasks:

- i. To measure and analyze the results of reviews and testing.
- ii. To monitor and document progress, test coverage and exit criteria.
- iii. To provide information on testing.
- iv. To initiate corrective actions.
- v. To make decisions.

2. Analysis and Design

Test analysis and Test Design has the following major tasks:

- i. To review the test basis. (The test basis is the information we need in order to start the test analysis and create our own test cases. Basically it's a documentation on which test cases are based, such as requirements, design specifications, product risk analysis, architecture and interfaces. We can use the test basis documents to understand what the system should do once built.)
- ii. To identify test conditions.
- iii. To design the tests.
- iv. To evaluate testability of the requirements and system.
- v. To design the test environment set-up and identify and required infrastructure and tools.



3. Implementation and Execution

- i. During test implementation and execution, take the test conditions into test cases and procedures and other test ware such as scripts for automation, the test environment and any other test infrastructure.
- ii. (Test cases is a set of conditions under which a tester will determine whether an application is working correctly or not.)
- iii. (Test ware is a term for all utilities that serve in combination for testing a software like scripts, the test environment and any other test infrastructure for later reuse.)

Test implementation has the following major task:

- i. To develop and prioritize our test cases by using techniques and create test data for those tests.
- ii. (In order to test a software application you need to enter some data for testing most of the features.
- iii. Any such specifically identified data which is used in tests is known as test data.)
- iv. We also write some instructions for carrying out the tests which is known as test procedures.
- v. We may also need to automate some tests using test harness and automated tests scripts.
- vi. (A test harness is a collection of software and test data for testing a program unit by running it under different conditions and monitoring its behavior and outputs.)
- vii. To create test suites from the test cases for efficient test execution.
- viii. (Test suite is a collection of test cases that are used to test a software program to show that it has some specified set of behaviors.
- ix. A test suite often contains detailed instructions and information for each collection of test cases on the system configuration to be used during testing.
- x. Test suites are used to group similar test cases together.)
- xi. To implement and verify the environment.

Test execution has the following major task:

- i. To execute test suites and individual test cases following the test procedures.
- ii. To re-execute the tests that previously failed in order to confirm a fix. This is known as confirmation testing or re-testing.
- iii. To log the outcome of the test execution and record the identities and versions of the software under tests.
- iv. The test log is used for the audit trial.
- v. (A test log is nothing but, what are the test cases that we executed, in what order we executed, who executed that test cases and what is the status of the test case (pass/fail).
- vi. These descriptions are documented and called as test log.)
- vii. To compare actual results with expected results.
- viii. Where there are differences between actual and expected results, it report discrepancies as Incidents.



4. Evaluating Exit criteria and Reporting
- i. Based on the risk assessment of the project we will set the criteria for each test level against which we will measure the “enough testing”. These criteria vary from project to project and are known as exit criteria.
 - ii. Exit criteria come into picture, when:
 1. Maximum test cases are executed with certain pass percentage.
 2. Bug rate falls below certain level.
 3. When achieved the deadlines.
 - iii. Evaluating exit criteria has the following major tasks
 1. To check the test logs against the exit criteria specified in test planning.
 2. To assess if more test are needed or if the exit criteria specified should be changed.
 3. To write a test summary report for stakeholders.
5. Test Closure activities:
Test closure activities are done when software is delivered. The testing can be closed for the other reasons also like:
1. When all the information has been gathered which are needed for the testing.
 2. When a project is cancelled.
 3. When some target is achieved.
 4. When a maintenance release or update is done.
- Test closure activities have the following major tasks:
1. To check which planned deliverables are actually delivered and to ensure that all incident reports have been resolved.
 2. To finalize and archive test ware such as scripts, test environments, etc. for later reuse.
 3. To handover the test ware to the maintenance organization. They will give support to the software.
 1. To evaluate how the testing went and learn lessons for future releases and projects.

(B) Attempt any ONE of the following :

06 Marks

(a) Describe V-model with labelled diagram.

6M

Ans: Verification:

1. It makes sure that the product is designed to deliver all functionality to the customer.
2. Verification is done at the starting of the development process. It includes reviews and meetings, walkthroughs, inspection, etc. to evaluate documents, plans, code, requirements and specifications.
3. It answers the questions like: Am I building the product right?
4. Am I accessing the data right (in the right place; in the right way).
5. It is a Low level activity
6. Performed during development on key art facts, like walkthroughs, reviews and

(Diagram:2 marks, Explanation:4 marks)



inspections, mentor feedback, training, checklists and standards.

7. Demonstration of consistency, completeness, and correctness of the software at each stage and between each stage of the development life cycle.

Validation:

1. Determining if the system complies with the requirements and performs functions for which it is intended and meets the organization's goals and user needs.
2. Validation is done at the end of the development process and takes place after verifications are completed.
3. It answers the question like: Am I building the right product?
4. Am I accessing the right data (in terms of the data required to satisfy the requirement).
5. It is a High level activity.
6. Performed after a work product is produced against established criteria ensuring that the product integrates correctly into the environment.
7. Determination of correctness of the final software product by a development project with respect to the user needs and requirements.

OR

Verification Phase:

1. Overall Business Requirement: In this first phase of the development cycle, the product requirements are understood from customer perspective. This phase involves detailed communication with the customer to understand his expectations and exact requirements. The acceptance test design planning is done at this stage as business requirements can be used as an input for acceptance testing.

2. Software Requirement: Once the product requirements are clearly known, the system can be designed. The system design comprises of understanding & detailing the complete hardware, software & communication set up for the product under development. System test plan is designed based on system design. Doing this at earlier stage leaves more time for actual test execution later.

3. High level design: High level specification are understood & designed in this phase. Usually more than one technical approach is proposed & based on the technical & financial feasibility, the final decision is taken. System design is broken down further into modules taking up different functionality.

4. Low level design: In this phase the detailed integral design for all the system modules is specified. It is important that the design is compatible with the other modules in the system & other external system. Components tests can be designed at this stage based on the internal module design,

5. Coding: The actual coding of the system modules designed in the design phase is taken up in the coding phase. The base suitable programming language is decided based on requirements. Coding is done based on the coding guidelines & standards.

Validation Phase:

1. Unit Testing: Unit testing designed in coding are executed on the code during this validation phase. This helps to eliminate bugs at an early stage.

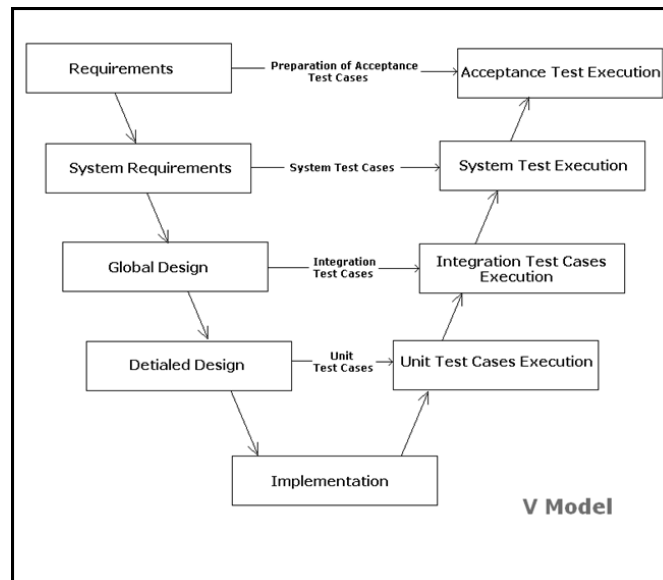


2. Components testing: This is associated with module design helps to eliminate defects in individual modules.

3. Integration Testing: It is associated with high level design phase & it is performed to test the coexistence & communication of the internal modules within the system

4. System Testing: It is associated with system design phase. It checks the entire system functionality & the communication of the system under development with external systems. Most of the software & hardware compatibility issues can be uncovered using system test execution.

5. Acceptance Testing: It is associated with overall & involves testing the product in user Environment. These tests uncover the compatibility issues with the other systems available in the user environment. It also uncovers the non-functional issues such as load & performance defects in the actual user environment.



(b) What is the use of code complexity testing? Also compute code complexity with the help of suitable example. 6M

Ans:

1. Program Statements and Line Coverage (Code Complexity Testing)

- The most straightforward form of code coverage is called statement coverage or line coverage.
- If you're monitoring statement coverage while you test your software, your goal is to make sure that you execute every statement in the program at least once.
- With line coverage the tester tests the code line by line giving the relevant output.

For example

- `#include<stdio.h>`
- `void main()`

(Explanation: 2 marks, Example: 2 marks)



```
3. {  
4. int i , fact= 1, n;  
5. printf("enter the number ");  
6. scanf("%d", &n);  
7. for(i =1 ;i <=n; i++)  
8. fact = fact * i;  
9. printf ("the factorial of a number is →"%d", fact);  
10. }
```

2. Branch Coverage (Code Complexity Testing)

- i. Attempting to cover all the paths in the software is called path testing.
- ii. The simplest form of path testing is called branch coverage testing.
- iii. To check all the possibilities of the boundary and the sub boundary conditions and it's branching on those values.
- iv. Test coverage criteria requires enough test cases such that each condition in a decision takes on all possible outcomes at least once, and each point of entry to a program or subroutine is invoked at least once.
- v. Every branch (decision) taken each way, true and false.
- vi. It helps in validating all the branches in the code making sure that no branch leads to abnormal behavior of the application.

3. Condition Coverage (Code Complexity Testing)

- i. Just when you thought you had it all figured out, there's yet another complication to path testing.
- ii. Condition coverage testing takes the extra conditions on the branch statements into account.

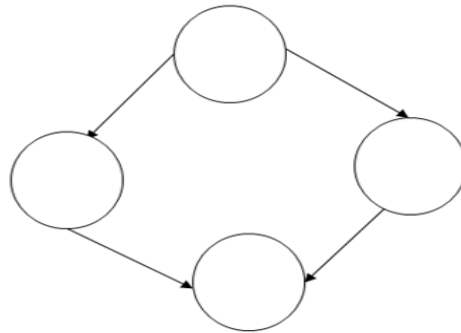
4. Code Complexity: Cyclomatic complexity is metric that quantifies the complexity of a program and provides answers to the questions such as



- 1. Which of the paths are independent? If two paths are not independent then we may be able to minimize the number of tests.
- 2. Is there an upper bound on the number of tests that must be run to ensure that all the statements have been executed at least once?
- In this a program is represented as flow graph. A flow graph consists of nodes and edges.
- Cyclomatic complexity=Number of predicate nodes(P)+1

Or

- Cyclomatic complexity=Edges(E)-Nodes(N)+2



In the above flow graph: No. of independent path=2, No. of edges $E=4$, No. of nodes $N=4$

Cyclomatic complexity= $E-N+2=4-4+2=2$ or

Cyclomatic complexity= $P+1=1+1=2$



5.		Attempt any TWO of the following :	16 Marks
	(a)	Describe Test Management Process and give details of following internal standards for process and method : (i) Naming and storage contention. (ii) Documentation standard.	8M
	Ans:	a) Test Management Process: Test Management: It concerned with both test resource and test environment management. It is the role of test management to ensure that new or modified service products meet business requirements for which they have been developed or enhanced. Test Management Process: 1. Test Plan: Test plan served as an initial sketch to carry out the testing. Testing is being tracked and monitored as per the test plan. It gives a prior picture of test challenge and aspect that will be carried out for the software. 2. Test design affords how to implement the testing. Typically creating test cases is with inputs and expected output of the system and choosing which test cases are necessary for the execution of the test. 3. Test Execution : Manner of executing and test the actual system result against the expected result is test execution. Test execution can be done manually and by using automation suit. During the execution tester needs to make sure, that the user's need of the software is occupied in the software. 4. Exit criteria determines when to stop the test execution. Exit criteria is defined during the test plan phase and used in the test execution phase as a mile stone. Tester needs to set the exit criteria at the beginning, exit criteria may change during the project run as well. 5. Test reporting gives the picture of test process and result for the particular testing cycle. To define the element in the test reporting the first thing that needs to be considered is whom the audiences of the test report are. For an example a project manager will like to see the high level picture of the testing, intermediate people will wish to view more detail and the client will expect the test reporting in the criteria such as requirement basis, feature basis.	(Test Management Process:2 marks, Naming Standard s:3marks, Documentation Standard s:3marks)



1. **Naming and storage conventions** for test artifacts: Every test artifacts(test specification, test case, test results and so on)have to be named appropriately and meaningfully. It enables

- a) Easy identification of the product functionality.
- b) Reverse mapping to identify the functionality corresponding to a given set of tests.

e.g. modules shall be M01,M02.Files types can be .sh, .SQL.
In addition to file naming conventions, the standards may also stipulate the conventions for directory structures for tests. These directory structures are mapped into configuration management repository.

2. **Documentation standards:** Documentation standards specify how to capture information about the tests within the test scripts themselves. It should include:

- a. Appropriate header level comments at the beginning of a file that outlines the functions to be served by the test.
- b. Sufficient inline comments, spread throughout the file
- c. Up-to-Date change history information, reading all the changes made to the test file.

(b) What is boundary value analysis? Explain with suitable example.

8M

Ans: Most of the defects in software products have around conditions and boundaries. By boundaries, we mean “limits” of values of the various variables. This is one of the software testing technique in which the test cases are designed to include values at the boundary. If the input data is used within the boundary value limits, then it is said to be Positive Testing. If the input data is picked outside the boundary value limits, then it is said to be Negative Testing. Boundary value analysis is another black box test design technique and it is used to find the errors at boundaries of input domain rather than finding those errors in the center of input. Each boundary has a valid boundary value and an invalid boundary value. Test cases are designed based on the both valid and invalid boundary values. Typically, we choose one test case from each boundary. Boundary value analysis help identify the test cases that are most likely to uncover defects.

(Explanation:5marks, Example:3 marks or any other relevant example)



MODEL ANSWER
SUMMER- 17 EXAMINATION

Subject Title: **SOFTWARE TESTING**

Subject Code: **17624**

Some examples of Boundary value analysis concept are: If the can accept he numbers between 1 to 100.Then
One test case for exact boundary values of input domains each means 1 and 100.
One test case for just below boundary value of input domains each means 0 and 99.
One test case for just above boundary values of input domains each means 2 and 101.
Under this technique, boundary values 0,1, 2, 99,100,101can be tested.
Out of which 1,2,99,100 are valid values and 0,101 are invalid values.

- (c) **Write a short note on :**
(i) Load testing
(ii) Stress testing
(iii)Recovery testing
(iv)Usability testing

8M

- Ans:**
- (i) Load testing:** Load is testing the software under customer expected load. In order to perform load testing on the software you feed it all that it can handle. Operate the software with largest possible data files. If the software operates on peripherals such as printer, or communication ports, connect as many as you can. If you are testing an internet server that can handle thousands of simultaneous connections, do it. With most software it is important for it to run over long periods. Some software's should be able to run forever without being restarted. So Time acts as a important variable.
- ii) Stress testing:** Stress testing is testing the software under less than ideal conditions. So subject your software to low memory, low disk space, slow cpus, and slow modems and so on. Look at your software and determine what external resources and dependencies it has. Stress testing is simply limiting them to bare minimum. With stress testing you starve the software. **For e.g.** Word processor software running on your computer with all available memory and disk space, it works fine. But if the system runs low on resources you had a greater potential to expect a bug. Setting the values to zero or near zero will make the software execute different path as it attempt to handle the tight constraint. Ideally the software would run without crashing or losing data.
- iii) Recovery testing:** Recovery testing is a type of non-functional testing. Recovery testing is done in order to check how fast and better the application can recover after it has gone through any type of crash or hardware failure etc.
- Recovery testing is the forced failure of the software in a variety of ways to verify that recovery is properly performed.
 - Determining the feasibility of the recovery process.
 - Verification of the backup facilities.
 - Ensuring proper steps are documented to verify the compatibility of backup facilities.
 - Providing Training within the team.

(Load testing:2 marks, Stress Testing:2 marks, Recovery Testing:2 marks, Usability Testing:2 marks)



- Demonstrating the ability of the organization to recover from all critical failures.
 - Maintaining and updating the recovery plan at regular intervals.
- (iv) Usability testing: Usability testing, a non-functional testing technique that is a measure of how easily the system can be used by end users. It is difficult to evaluate and measure but can be evaluated based on the below parameters:
- Levels of Skill required learn/use the software. It should maintain the balance for both novice and expert user.
 - Time required to get used to in using the software.
 - The measure of increase in user productivity if any.
 - Assessment of a user's attitude towards using the software.
 - Usability testing, a non-functional testing technique that is a measure of how easily the system can be used by end users.
 - It is difficult to evaluate and measure but can be evaluated based on the below parameters:
 - Levels of Skill required learn/use the software. It should maintain the balance for both novice and expert user.
 - Time required to get used to in using the software.
 - The measure of increase in user productivity if any.

6.

Attempt any FOUR of the following :

16 Marks

(a)

Describe quality assurance and quality control.

4M

Ans:

Quality Assurance:

- i. It is Process oriented activities.
- ii. A part of quality management focused on providing confidence that quality requirements will be fulfilled.
- iii. All the planned and systematic activities implemented within the quality system that can be demonstrated to provide confidence that a product or service will fulfill requirements for quality
- iv. Quality Assurance is fundamentally focused on planning and documenting those processes to assure quality including things such as quality plans and inspection and test plans.
- v. Quality Assurance is a system for evaluating performance, service, of the quality of a product against a system, standard or specified requirement for customers.
- vi. Quality Assurance is a complete system to assure the quality of products or services. It is not only a process, but a complete system including also control. It is a way of management.

(Quality assurance: 2 marks, Quality Control: 2 marks)



MODEL ANSWER

SUMMER- 17 EXAMINATION

Subject Title: SOFTWARE TESTING

Subject Code:

17624

Quality Control:

- i. It is Product oriented activities.
- ii. A part of quality management focused on fulfilling quality requirements.
- iii. The operational techniques and activities used to fulfill requirements for quality.
- iv. Quality Control on the other hand is the physical verification that the product conforms to these planned arrangements by inspection, measurement etc.
- v. Quality Control is the process involved within the system to ensure job management, competence and performance during the manufacturing of the product or service to ensure it meets the quality plan as designed.
- vi. Quality Control just measures and determines the quality level of products or services.

(b) Describe alpha testing with its entry & exit criteria.

4M

Ans:

Alpha testing is done by the customers in controlled environment in front of the development team. It has less probability of finding errors. It is done during implementation phase of software.

(Alpha testing:2 marks, Entry & exit Criteria:2 marks)

When to Start and Stop Testing of Software (Entry and Exit Criteria)

Process model is a way to represent any given phase of software development that prevent and minimize the delay between defect injection and defect detection/correction.

- **Entry criteria**, specifies when that phase can be started also included the inputs for the phase.
- Tasks or steps that need to be carried out in that phase, along with measurements that characterize the tasks.
- Verification, which specifies methods of checking that tasks have been carried out correctly.
- Clear entry criteria make sure that a given phase does not start prematurely.
- The verification for each phase helps to prevent defects. At least defects can be minimized.

Exit criteria, which stipulate the conditions under which one can consider the phases as done and included are the outputs for the phase.

a. Exit criteria may include:

1. All test plans have been run
2. All requirements coverage has been achieved.
3. All severe bugs are resolved.

OR

Entry Criteria for Alpha testing:

- Software requirements document or Business requirements specification
- Test Cases for all the requirements
- Testing Team with good knowledge about the software application



- Test Lab environment setup
- QA Build ready for execution
- Test Management tool for uploading test cases and logging defects
- Traceability Matrix to ensure that each design requirement has atleast one test case that verifies it

Exit Criteria for Alpha testing:

- All the test cases have been executed and passed.
- All severity issues need to be fixed and closed
- Delivery of Test summary report
- Make sure that no more additional features can be included
- Sign off on Alpha testing

(c) What are types of test report? Write contents of test summary report.

4M

Ans: Test reporting is a means of achieving communication through the testing cycle. There are 3 types of test reporting.

1. Test incident report: A test incident report is communication that happens through the testing cycle as and when Defects are encountered .A test incident report is an entry made in the defect repository each defect has a unique id to identify incident .The high impact test incident are Highlighted in the test summary report.

2. Test cycle report: A test cycle entails planning and running certain test in cycle, each cycle using a different build of the product .As the product progresses through the various cycles it is expected to stabilize.

Test cycle report gives :

1. A summary of the activities carried out during that cycle.
2. Defects that are uncovered during that cycle based on severity and impact
3. Progress from the previous cycle to the current cycle in terms of defect fixed
4. Outstanding defects that not yet to be fixed in cycle
5. Any variation observed in effort or schedule

3. Test summary report: The final step in a test cycle is to recommend the suitability of a product for release. A report that summarizes the result of a test cycle is the test summary report. **There are two types of test summary report:**

1. Phase wise test summary ,which is produced at the end of every phase
2. Final test summary report.

A Summary report should content:

1. Test Summary report Identifier
2. Description : Identify the test items being reported in this report with test id
3. Variances: Mention any deviation from test plans, test procedures, if any.
4. Summary of results: All the results are mentioned here with the resolved incidents and their solutions.
5. Comprehensive assessment and recommendation for release should include Fit for release assessment and recommendation of release

(Types of test Report:2 marks, Content of test Summary Report:2 marks)



(d)	List all defect classification. Also describe any one defect in brief.	4M
Ans:	<p>List of Defect Classification:</p> <ol style="list-style-type: none"> 1. Requirements and specification defect 2. Design Defects 3. Coding Defects 4. Testing Defect <p>Requirements and specification defect: Requirement related defects arise in a product when one fails to understand what is required by the customer. These defects may be due to customer gap, where the customer is unable to define his requirements, or producer gap, where developing team is not able to make a product as per requirements. Defects injected in early phases can persist and be very difficult to remove in later phases. Since any requirements documents are written using natural language representation, there are very often occurrences of ambiguous, contradictory, unclear, redundant and imprecise requirements. Specifications are also developed using natural language representations.</p> <p>Design Defects: Design defects occur when system components, interactions between system components, interactions between the outside software/hardware, or users are incorrectly designed. This covers in the design of algorithms, control, logic/ data elements, module interface descriptions and external software/hardware/user interface descriptions. Design defects generally refer to the way of design creation or its usage while creating a product. The customer may or may not be in a position to understand these defects, if structures are not correct. They may be due to problems with design creation and implementation during software development life cycle.</p> <p>Coding Defects: Coding defects may arise when designs are implemented wrongly. If there is absence of development/coding standards or if they are wrong, it may lead to coding defects. Coding defects are derived from errors in implementing the code. Coding defect classes are closely related to design defect classes especially if pseudo code has been used for detailed design. Some coding defects come from a failure to understand programming language constructs, and miscommunication with the designers. Others may have transcription or omission origins. At times it may be difficult to classify a defect as a design or as a coding defect.</p> <p>Testing Defect: Testing defect are defects introduced in an application due to wrong testing, or defects in the test artifact leading to wrong testing. Defects which cannot be reproduced, or are not supported by requirement or are duplicate may represent a false call. In this defects includes</p> <ol style="list-style-type: none"> 1. Test-design defect: test-design defect refers to defects in test artifacts. There can be defects in test plans, test scenarios, test cases and test data definition which can lead to defect in software. 2. Test-environment defect: this defect may arise when test environment is not set properly. Test environment may be comprised of hardware, software, simulator and people doing 	(List of Classification: 2 marks, Description of any One: 2 marks)



testing.

3. Test-tool defects: any defects introduced by a test tool may be very difficult to find and resolve, as one may have to find the defect using manual test as against automated tools.

OR

Software Defects/ Bugs are normally classified as per:

- Severity / Impact
- Probability / Visibility
- Priority / Urgency
- Related Dimension of Quality
- Related Module / Component
- Phase Detected
- Phase Injected

(e) List & explain techniques of finding bugs.

4M

Ans:

List of techniques:

- 1.Static testing
- 2.Dynamic testing
- 3.Operational testing

Static Techniques: Static techniques of quality control define checking the software product and related artifacts without executing them. It is also termed desk checking/verification /white box testing'. It may include reviews, walkthroughs, inspection, and audits Here; the work product is reviewed by the reviewer with the help of a checklist, standards, any other artifact, knowledge and experience, in order to locate the defect with respect to the established criteria. Static technique is so named because it involves no execution of code, product, documentation, etc. This technique helps in establishing conformance to requirements view.

Dynamic Testing: Dynamic testing is a validation technique which includes dummy or actual execution of work products to evaluate it with expected behavior. It includes black box testing methodology such as system testing and unit testing. The testing methods evaluate the product with respect to requirements defined, designs created and mark it as pass or fail'. This technique establishes fitness for use'view.

Operational techniques: Operational techniques typically include auditing work products and projects to understand whether the processes defined for development /testing are being followed correctly o not, and also whether they are effective or not. It also includes revisiting the defects before and after fixing and analysis. Operational technique may include smoke testing and sanity testing of a work product.

OR

Techniques to find defects are:

a) Quick Attacks: The quick-attacks technique allows you to perform a cursory analysis of a system in a very compressed timeframe. Even without a specification, you know a little bit about the software, so the time spent is also time invested in developing expertise. The skill is relatively easy to learn, and once you've attained some mastery your quick-attack session

**(Listing:1
mark,
Explanati
on:3
marks)**



will probably produce a few bugs. Finally, quick attacks are quick. They can help you to make a rapid assessment. You may not know the requirements, but if your attacks yielded a lot of bugs, the programmers probably aren't thinking about exceptional conditions, and it's also likely that they made mistakes in the main functionality. If your attacks don't yield any defects, you may have some confidence in the general, happy-path functionality

b) Equivalence and Boundary Conditions: Boundaries and equivalence classes give us a technique to reduce an infinite test set into something manageable. They also provide a mechanism for us to show that the requirements are "covered".

c) Common Failure Modes: The heart of this method is to figure out what failures are common for the platform, the project, or the team; then try that test again on this build. If your team is new, or you haven't previously tracked bugs, you can still write down defects that "feel" recurring as they occur and start checking for them.

d) State-Transition Diagrams: Mapping out the application provides a list of immediate, powerful test ideas. Model can be improved by collaborating with the whole team to find "hidden" states transitions that might be known only by the original programmer or specification author. Once you have the map, you can have other people draw their own diagrams, and then compare theirs to yours. The differences in those maps can indicate gaps in the requirements, defects in the software, or at least different expectations among team members. e) Use Cases and Soap Opera Tests: Use cases and scenarios focus on software in its role to enable a human being to do something. Use cases and scenarios tend to resonate with business customers, and if done as part of the requirement process, they sort of magically generate test cases from the requirements. They make sense and can provide a straightforward set of confirmatory tests. Soap opera tests offer more power, and they can combine many test types into one execution.

f) Code-Based Coverage Models: Imagine that you have a black-box recorder that writes down every single line of code as it executes. Programmers love code coverage. It allows them to attach a number an actual, hard, real number, such as 75% to the performance of their unit tests, and they can challenge themselves to improve the score. Meanwhile, looking at the code that isn't covered also can yield opportunities for improvement and bugs.

g) Regression and High-Volume Test Techniques: People spend a lot of money on regression testing, taking the old test ideas described above and rerunning them over and over. This is generally done with either expensive users or very expensive programmers spending a lot of time writing and later maintaining those automated tests.