17624

11819

3 Hours / 100 Marks Seat No. | | | | | | | | |

Instructions : (1) All Questions are *compulsory.*

(2) Answer each next main Question on a new page.

(3) Illustrate your answers with neat sketches wherever necessary.

(4) Figures to the right indicate full marks.

(5) Assume suitable data, if necessary.

(6) Mobile Phone, Pager and any other Electronic Communication devices are not permissible in Examination Hall.

Marks

1. (A) Attempt any THREE of the following : **12**

(a) List and describe any four skills of software tester.

(b) Give any four difference between walkthroughs and inspections.

(c) With the help of diagram, describe bidirectional integration testing.

(d) What is test deliverables and milestones ? Give any four types of test deliverables.

(B) Attempt any ONE of the following : **6**

(a) Explain client-server testing with an example.

(b) Explain defect classification with an example.

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

(a) What is user documentation testing ? State any three objectives of user documentation testing.

(b) What are different static and dynamic testing tools ? Explain any one tool in detail.

(c) Describe acceptance testing with its advantages.

(d) Enlist factors considered for selecting a testing tool for test automation.

(e) Explain defect management process with proper diagram.

(f) Explain requirement based testing with it's stages and requirement testing process.

3. Attempt any FOUR of the following : **16**

(a) Draw classification of white box testing. Explain the purpose of code coverage testing.

(b) Give any four difference between smoke and sanity testing.

(c) Write four test cases for railway reservation system.

(d) Describe unit testing with suitable diagram.

(e) Define the term metrics and measurements. Explain need of software measurement.

4. (A) Attempt any THREE of the following : **12**

(a) What are the contents of 'Test Summary Report' used in test reporting ?

(b) State any eight limitations of manual testing.

(c) Explain defect Report Template with it's attributes.

(d) What are different features of test strategies ? Describe any one feature.

(B) Attempt any ONE of the following : **6**

(a) Explain V-Model with labelled diagram. State it's any two advantages and disadvantages.

(b) Give any six difference between white box testing and black box testing.

5. Attempt any TWO of the following : **16**

(a) Explain in detail, how to prepare a test plan with suitable example. Also, explain risk management during test planning.

(b) With the help of suitable example, explain the use of decision table in the black box testing.

(c) Differentiate between alpha testing and beta testing with parameters.

6. Attempt any FOUR of the following : **16**

(a) What are different techniques for finding defects ? Explain any one technique with an example.

(b) What is test case ? Which parameters are to be considered while documenting test cases ?

(c) Explain defect prevention cycle with neat diagram.

(d) List and explain any two types of standards in test management.

(e) Explain performance testing with an example.

WINTER– 18 EXAMINATION

Subject Name: Software Testing **Model Answer** 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 M**
	a	**List and describe any four skills of software tester.**	4 M
	Ans	Following are the various skill required to become a good software tester: · **Communication Skills:** Testers are expected to be good listeners as well as good presenters. A good software tester must have strong verbal and written communication skills. They require a good communication with developers before, during and after the project. The test reports, plans/cases which testers made should be easy to read and comprehend. They must be good listeners, good speakers, good writers, and good readers etc. all at the same time. Communication skills of a good software tester include his/her body language, their words, tone, writing styles, listening and attending others etc. · **Domain Knowledge:** Testers should have the detailed knowledge about the software or application, whether they are not domain experts and this knowledge will help them to find such errors which a user can face while using the application. After testing the application/software, the tester should keep the domain in his/her mind while arranging the errors in order according to their priority. As the testers may be working in different domain, on different technologies so they should be aware of any challenges and complexities. And the tester which have this domain knowledge quality may also	Listing 1M and Description 3M

include a good user interface, able to differentiate between a trivial and critical issue and also the better understanding of issues as they are some pros of Domain Knowledge.

· **Desire to Learn:** Testers should have the brief knowledge regarding the latest technologies, tools & techniques and they can also use them during testing. As there are various tools and techniques for development as well as testing & every tool or technique has some positive and negative effects. Testers must be able to learn new technologies & can also use them while testing. Working with latest tools/techniques might be difficult for tester but they can get something new.

· **Technical Skills:** A good software tester must have strong Technical skills. They must have proper knowledge about the coding skills in order to understand the application, good communication with developers and write test automation. The technical skills also include high proficiency in tools like MS Office, testing tools etc. These skills can be obtained by practicing and proper training.

· **Analytical Skill:** A good software tester should have to be able to check out how to reproduce the errors because only finding errors are not sufficient. For better understanding and creating of good test reports, analytical skill will break the complex software system into smaller units. Testers should have to analyze the situation of user while using the software or application. Testing report is a SWOT i.e. 'Strength', 'Weakness', 'Opportunity' and 'Threat' analysis of Software.

· **Planning:** First of all testers should have to plan how to make the testing report. Testing should be in proper manner i.e. it should cover all the functionalists, requirements, features and critical aspects of business etc. The testing report should be made in an exact order i.e. according to the priority of the errors/bugs. For better judgment of testing report good planning is very necessary.

· **Integrity:** Testers find the errors in applications/software with an assurance that developers will absolutely fix them. The testing report should have to show the priorities of the errors i.e. the report should be in various levels according to their priority level.

· **Curiosity:** During testing or analyzing any software, the testers must know about various applications, various domains etc. As domain has its own specialty so testers must have the curiosity to understand the domain under testing. They should have an eagerness of understanding the complexity and expectations.

· **Think from Users Perspective:** Each and every product is developed and designed for customers. Customers may not be having all technical skills a tester is having, if you fail to keep this in mind you might miss important bugs.

· **Be a Good Judge of Your Product:** Last but not the least; you have to be a good judge of your product. Ask yourself questions whether the software meeting all the requirements it should be having. As now you are the judge and you have the powers to distinguish between right and wrong. Judge listens to all testers in team.

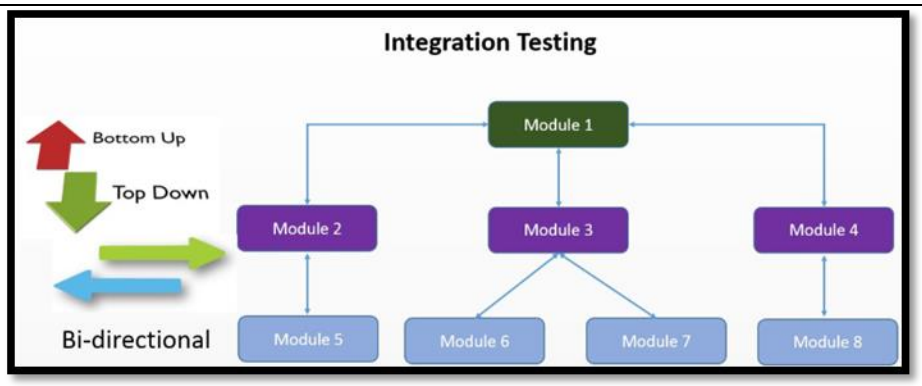
b **Give any four differences between walkthroughs and inspections.** — 4 M

Ans — 1M for each point

Walkthrough	Inspection
Walkthrough is informal	Inspection is formal
It is initiated by the author	It is initiated by the project team
Walkthrough is unplanned.	It is planned meeting with fixed roles assigned to all the members involved
Author reads the product code and his team mate comes up with defects or suggestions	Reader reads the product code. Everyone inspects it and comes up with defects.
Author makes a note of defects and suggestions offered by team mate	Recorder records the defects
It is informal, so there is no moderator	Moderator has a role in making sure that the discussions proceed on the productive lines

c **With the help of diagram, describe bidirectional integration testing.** — 4 M

Ans — 2M for diagram and 2M for explanation



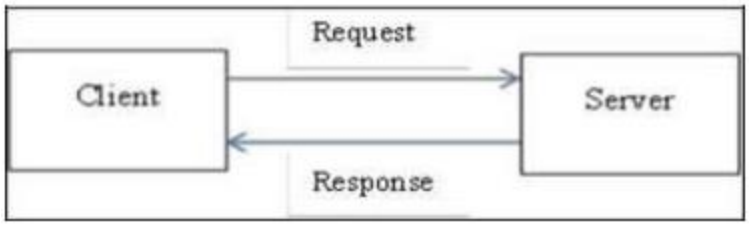
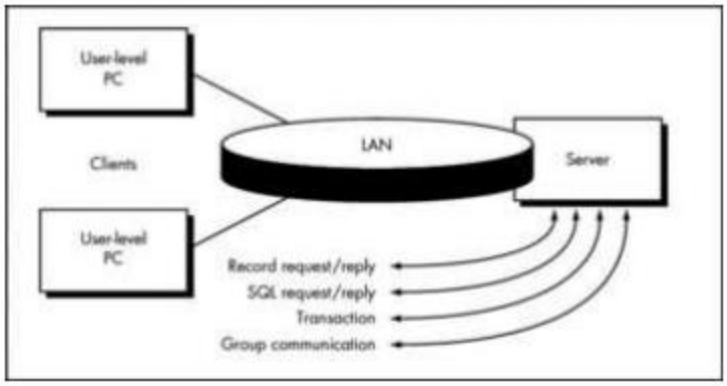
1. Bi-directional Integration is a kind of integration testing process that combines top-down and bottom-up testing.

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.	
	d	**What are test deliverables and milestones? Give any four types of test deliverables.**	4 M
	Ans	Test Deliverables are the artifacts which are given to the stakeholders of software project during the software development lifecycle. There are different test deliverables at every phase of the software development lifecycle. Some test deliverables are provided before testing phase, some are provided during the testing phase and some after the testing cycles is over. The different types of Test deliverables are: Test cases Documents Test Plan Testing Strategy Test Scripts Test Data Test Traceability Matrix Test Results/reports Test summary report Install/config guides Defect Reports Release notes 1. The test plan describes the overall method to be used to verify that the software meets the product specification and the customer's needs. It includes the quality objectives, resource needs, schedules, assignments, methods, and so forth. 2. Test cases list the specific items that will be tested and describe the detailed steps that will be followed to verify the software. 3. Bug reports describe the problems found as the test cases are followed. These could	Test Deliverables and Milestone – 1M each. Types of test deliverables 2M

		be done on paper but are often tracked in a database. 4. Test tools and automation are listed and described which are used to test the software. If the team is using automated methods to test software, the tools used, either purchased or written in-house, must be documented. 5. Metrics, statistics, and summaries convey the progress being made as the test work progresses. They take the form of graphs, charts, and written reports. **Milestones:** Milestones are the dates of completion given for various tasks to be performed in testing. These are thoroughly tracked by the test manager and are kept in the documents such as Gantt charts, etc.	
	B	**Attempt any ONE of the following :**	**6 M**
	a	**Explain client-server testing with example.**	6 M
	Ans	  In Client-server testing there are several clients communicating with the server. 1. Multiple users can access the system at a time and they can communicate with the server. 2. Configuration of client is known to the server with certainty. 3. Client and server are connected by real connection. 4. Testing approaches of client server system: **1. Component Testing:** One need to define the approach and test plan for testing client and server individually. When server is tested there is need of a client simulator,	2M for diagram and 2 M for ex[planation

		whereas testing client a server simulator, and to test network both simulators are used at a time. **2. Integration testing:** After successful testing of server, client and network, they are brought together to form system testing. **3. Performance testing:** System performance is tested when numbers of clients are communicating with server at a time. Volume testing and stress testing may be used for testing, to test under maximum load as well as normal load expected. Various interactions may be used for stress testing. **4. Concurrency Testing:** It is very important testing for client-server architecture. It may be possible that multiple users may be accessing same record at a time, and concurrency testing is required to understand the behavior of a system in this situation. **5. Disaster Recovery/ Business continuity testing:** When the client server are communicating with each other, there exit a possibility of breaking of the communication due to various reasons or failure of either client or server or link connecting them. The requirement specifications must describe the possible expectations in case of any failure. **6. Testing for extended periods:** In case of client server applications generally server is never shutdown unless there is some agreed Service Level Agreement (SLA) where server may be shut down for maintenance. It may be expected that server is running 24X7 for extended period. One needs to conduct testing overran extended period to understand if service level of network and server deteriorates over time due to some reasons like memory leakage. **7. Compatibility Testing:** Client server may be put in different environments when the users are using them in production. Servers may be in different hardware, software, or operating system environment than the recommended. Other testing such as security testing and compliance testing may be involved if needed, as per testing and type of system. **E.g.:** applications developed in VB, VC++, Core Java, C, C++, D2K, PowerBuilder etc., The backend for these applications would be MS Access, SQL Server, Oracle, Sybase, Mysql, Quadbase.	
	b	**Explain defect classification with an example.**	6 M
	Ans	Defect Classification: **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	3M for explanation and 1M for example

		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. **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. **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 detect. **Testing Defects:** 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 : **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 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. Example: For flight operating website, defect in generating the ticket number against reservation is high severity and also high priority.	
2		**Attempt any FOUR of the following :**	**16 M**
	a	**What is user documentation testing? State any three objectives of user**	4 M

		documentation testing.	
	Ans	• User documentation covers all the manuals, user guides, installation guides, setup guides, read me file software release notes and online help that are provided along with the software to help the end user to understand the software system. • Documentation is as important to a product's success as the product itself. If the documentation is poor, non-existent, or wrong, it reflects on the quality of the product and the vendor. • As per the IEEE Documentation describing plans for, or results of, the testing of a system or component, Types include test case specification, test incident report, test log, test plan, test procedure, test report. Hence the testing of all the above mentioned documents is known as documentation testing. • Objectives of user documentation testing's are: ➢ To check if what is stated in the documentation is available in the product. ➢ To check if what is there in the product is explained correctly in the document. ➢ To check if all the requirements of the user have been fulfilled correctly.	2M for explanation and 2 M for objectives – Any other relevant objectives shall be given marks
	b	**What are different static and dynamic testing tools? Explain any one tool in detail.**	4 M
	Ans	**Static testing tools:** Static testing tools are used during static analysis of a system. □ Static testing tools are used throughout a software development life cycle, e.g., tools used for verification purposes. There are many varieties of static testing tools used by different people as per the type of system being developed. Code complexity measurement tools can be used to measure the complexity of a given code. Similarly, data-profiling tools can be used to optimize a database. Code-profiling tools can be used to optimize code. Test-generators are used for generating a test plan form code. Syntax-checking tools are used to verify correctness of code. **Dynamic testing tools:** Dynamic testing tools are used at different levels of testing starting from unit testing & which may go up to system testing & performance testing. These tools are generally used by tester. There are many different tools used for dynamic testing. Some of the areas covered by testing tools are: 1. Regression testing using automated tools. 2. Defect tracking and communication systems used by tracking & communication. 3. Performance, Load, stress-testing tools. Regression testing with automated tools: Testing Whiz's -**regression testing automation** solution helps you determine obstructed functions and features for specific releases ensuring minimum risks into production and maximum test coverage.	2M for static and dynamic tool explanation and 2M for explanation of any tool

	c	**Describe acceptance testing with its advantages.**	4 M
	Ans	• 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. **Advantages:** • It is conducted to ensure that system requirements meet business needs. • The UAT process allows for any issues to be fixed before the system goes live. • It helps in simulating the real-time user behavior and environment. • It allows the company to improve the software quality by involving customer feedback.	2M for explanation and 2 M for advantages
	d	**Enlist factors considered for selecting a testing tool for test automation.**	4 M
	Ans	Criteria for Selecting Test Tools: The Categories for selecting Test Tools are, 1. Meeting requirements; 2. Technology expectations; 3. Training/skills; 4. Management aspects. 1. Meeting requirements- There are plenty of tools available in the market but rarely do they meet all the requirements of a given product or a given organization. Evaluating different tools for different requirements involve significant effort, money, and time. Given of the plethora of choice available, huge delay is involved in selecting and implementing test tools. 2. Technology expectations- Test tools in general may not allow test developers to extends/modify the functionality of the framework. So extending the functionality requires going back to the tool vendor and involves additional cost and effort. A good number of test tools require their libraries to be linked with product binaries. 3. Training/skills- While test tools require plenty of training, very few vendors provide the training to the required level. Organization level training is needed to deploy the test tools, as the user of the test suite are not only the test team but also the development team and other areas like configuration management. 4. Management aspects- A test tool increases the system requirement and requires the	1M for each factor

hardware and software to be upgraded. This increases the cost of the already- expensive test tool.

OR

Guidelines for selecting a tool:

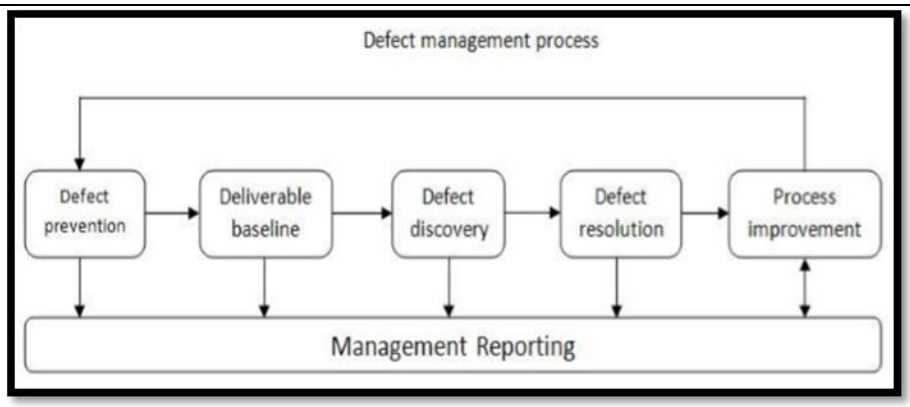
1. The tool must match its intended use. Wrong selection of a tool can lead to problems like lower efficiency and effectiveness of testing may be lost.

2. Different phases of a life cycle have different quality-factor requirements. Tools required at each stage may differ significantly.

3. Matching a tool with the skills of testers is also essential. If the testers do not have proper training and skill then they may not be able to work effectively.

4. Select affordable tools. Cost and benefits of various tools must be compared before making final decision.

5. Backdoor entry of tools must be prevented. Unauthorized entry results into failure of tool and creates a negative environment for new tool introduction.

	e	**Explain defect management process with proper diagram.**	4 M
	Ans		2M for diagram, 2M for explanation

1. Defect Prevention: Implementation of techniques, methodology and standard processes to reduce the risk of defects.

2. Deliverable Baseline: Deliverables are considered to be ready for further development. i.e. the deliverables meet exit criteria.

3. Defect Discovery: To find the defect through the process of verification and validation.

4. Defect Resolution: Defect is corrected or corrective action is taken and notification is given to tester.

5. Process Improvement: To identify ways to improve the process to prevent further future occurrences of similar defects i.e. Corrective and preventive action is taken for

		processes improvement. 6. Management Reporting: Reporting is about status of application and processes.	
	f	**Explain requirement based testing with its stages and requirement testing process.**	4 M
	Ans	Requirements-based testing is a testing approach in which test cases, conditions and data are derived from requirements. It includes functional tests and also non-functional attributes such as performance, reliability or usability. Stages in Requirements based Testing: • **Defining Test Completion Criteria** - Testing is completed only when all the functional and non-functional testing is complete. • **Design Test Cases** - A Test case has five parameters namely the initial state or precondition, data setup, the inputs, expected outcomes and actual outcomes. • **Execute Tests** - Execute the test cases against the system under test and document the results. • **Verify Test Results** - Verify if the expected and actual results match each other. • **Verify Test Coverage** - Verify if the tests cover both functional and non-functional aspects of the requirement. • **Track and Manage Defects** - Any defects detected during the testing process goes through the defect life cycle and are tracked to resolution. Defect Statistics are maintained which will give us the overall status of the project. Requirements Testing process: • Testing must be carried out in a timely manner. • Testing process should add value to the software life cycle, hence it needs to be effective. • Testing the system exhaustively is impossible hence the testing process needs to be efficient as well. • Testing must provide the overall status of the project, hence it should be manageable.	1M for definition, 2M for stages and 1 M for process
3		**Attempt any FOUR of the following :**	**16 M**
	a	**Draw classification of white box testing. Explain the purpose of code coverage testing**	4 M

Ans

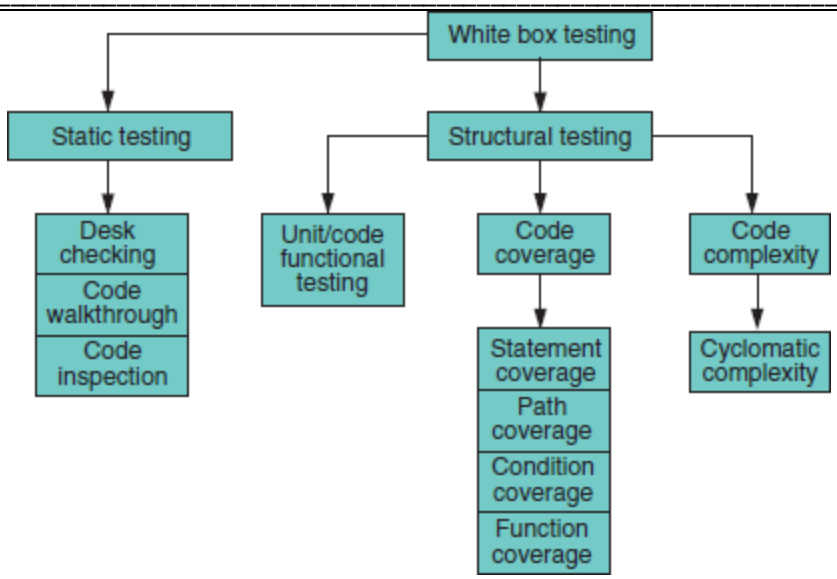


Diagram of classification of white box testing: 2 marks, Code coverage testing technique: 2 marks

Code coverage testing purpose:

i. The logical approach is to divide the code just as you did in black-box testing into its data and its states (or program flow).

ii. By looking at the software from the same perspective, you can more easily map the white-box information you gain to the black-box cases you've already written.

iii. Consider the data first. Data includes all the variables, constants, arrays, data structures, keyboard and mouse input, files and screen input and output, and

I/O to other devices such as modems, networks, and so on.
For example

```
1. #include<stdio.h>

2. void main()

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. }
```

The declaration of data is complete with the assignment statement and the variable declaration statements. All the variable declared are properly utilized.

b **Give any four differences between smoke and sanity testing.** 4 M

Ans

Smoke Testing	Sanity Testing
Smoke Testing is performed to ascertain that the critical functionalities of the program is working fine	Sanity Testing is done to check the new functionality / bugs have been fixed
The objective of this testing is to verify the "stability" of the system in order to proceed with more rigorous testing	The objective of the testing is to verify the "rationality" of the system in order to proceed with more rigorous testing
This testing is performed by the developers or testers	Sanity testing is usually performed by testers
Smoke testing is usually documented or scripted	Sanity testing is usually not documented and is unscripted
Smoke testing is a subset of Regression testing	Sanity testing is a subset of Acceptance testing
Smoke testing exercises the entire system from end to end	Sanity testing exercises only the particular component of the entire system
Smoke testing is like General Health Check Up	Sanity Testing is like specialized health check up

Any 4 valid points of comparison :4 marks, 1 mark each

c **Write four test cases for Railway reservation system.** 4 M

Ans

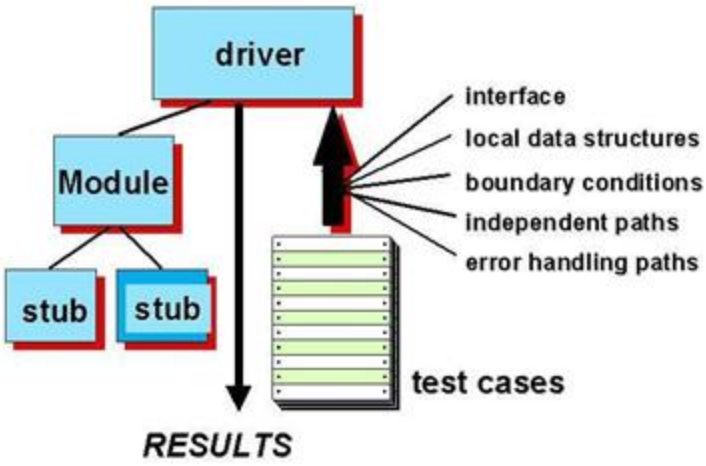
Test Case Id	Test case Objectives	Input Data	Expected Result	Actual Result	Status
TC1	Login field	Any valid login name (abcd xyz)	It should accept the login name	The login name is accepted	Pass
TC2	Password field	Valid Password	Valid password should be accepted	Successful login message	Pass
TC3	Password field	Invalid Password	Should not accept wrong password	Message displayed as Invalid login or wrong Password	Pass
TC4	Date of journey	Date format not before the current date	It should accept	Accepted	Pass
TC5	Date of return journey	Date greater than the return journey date	It should accept date	Accepted	Pass

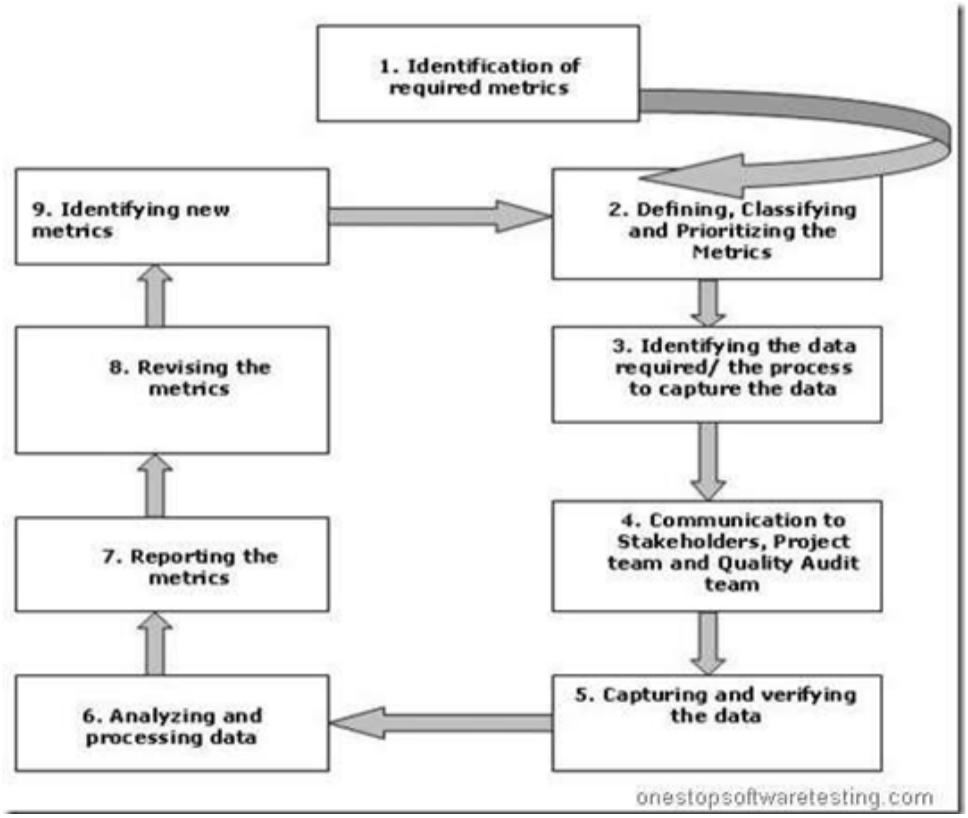
Any 4 valid test cases :4 marks, 1 mark each
Any other relevant test Cases shall be given marks.

d **Describe unit testing with suitable example.** 4 M

Ans Unit Testing: Software product is made up of many units , each unit needed to be tested to find whether they have implemented the design correctly or not.

Unit testing description: 4

		Additional Requirements : The module under consideration might be getting inputs from another module or the module is calling some another module . Some interface modules has to be simulated if required like drivers and stubs. Drivers: The module where the required inputs for the module under test are simulated for the purpose of module or unit testing is known as a Driver module. The driver module may print or interpret the result produced by the module under test. Stubs: The module under testing may also call some other module which is not ready at the time of testing. There is need of dummy modules required to simulate for testing, instead of actual modules. These are called stubs.  For example if a developer is developing a loop for searching functionality of an application which is a very small unit of the whole code of that application then to verify that the particular loop is working properly or not is known as unit testing.	marks, Diagram optional
	e	**Define the terms metrics and measurements. Explain need of software measurement.**	4 M
	Ans	A Metric is a measurement of the degree that any attribute belongs to a system, product or process. For example the number of errors per person hours would be a metric. Thus, software measurement gives rise to software metrics. A measurement is an indication of the size, quantity, amount or dimension of a particular attribute of a product or process. For example the number of errors in a system is a measurement. Software measurement is required to: Establish the quality of the current product or process. To predict future qualities of the product or process. To improve the quality of a product or process. To determine the state of the project in relation to budget and schedule.	Definition: 1 mark, Needs: 3 marks

			
4	**A**	**Attempt any THREE of the following :**	**12 M**
	a	**What are the contents of 'Test Summary Report' used in test reporting?**	4 M
	Ans	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 contain: 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 **OR**	Test summary report explanation : 4 marks

		• A test report is any descriptions, explanation or justification the status of a test project. • A comprehensive test report is all of those things together. • Test reporting is a means of achieving this communication. Types of reports that might be generated for a classical waterfall based project include: • summary report for the phase; • technical review report; • walkthrough report; • Audit report. • test report Summary Report: • The phase may be between five and ten pages in length. • It would cover the major points drawn out from the testing. • It is suggested that the report be structured with a single highlight sheet at the front based on a small commentary and a table that will show key areas where successful testing has been completed and areas where concern must be documented. • The report should also address any issues that arose from external dependencies that may have affected the schedule.	
	b	**State any eight limitations of manual testing.**	4 M
	Ans	**Manual Testing:** Testing Computer Software manually without using any Automation Tools **Limitations of Manual Testing**: • Manual Testing requires more time or more resources, sometimes both • Performance testing is impractical in manual testing. • Less Accuracy • Executing same tests again and again time taking process as well as Tedious. • GUI Objects Size difference and Color combinations etc. Are not easy to find in Manual Testing. • Not Suitable for Large scale projects and time bounded projects. • Batch Testing is not possible, for each and every test execution Human user Interaction is mandatory. • Manual Test Case scope is very limited, if it is Automated test, scope is unlimited. • Comparing large amount of data is impractical • Checking relevance of search of operation is difficult • Processing change requests during software maintenance takes more time.	Any 8 limitations :1/2 mark each
	c	**Explain defect report template with its attributes.**	4 M

Ans

DEFECT REPORT TEMPLATE

In most companies, a defect reporting tool is used and the elements of a report can vary. However, in general, a defect report can consist of the following elements.

ID	Unique identifier given to the defect. (Usually Automated)
Project	Project name.
Product	Product name.
Release Version	Release version of the product. (e.g. 1.2.3)
Module	Specific module of the product where the defect was detected.
Detected Build Version	Build version of the product where the defect was detected (e.g. 1.2.3.5)
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. Keep it simple but comprehensive.
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.
Attachments	Attach any additional information like screenshots and logs.
Remarks	Any additional comments on the defect.
Defect Severity	Severity of the Defect.

A defect report documents an anomaly discovered during testing. It includes all the information needed to reproduce the problem, including the author, release/build number, open/close dates, problem area, problem description, test environment, defect type, how it was detected, who detected it, priority, severity, status, etc.

After uncovering a defect (bug), testers generate a formal defect report. The purpose of a defect report is to state the problem as clearly as possible so that developers can replicate the defect easily and fix it.

Defect Report Template & its explanation: 4 marks

d **What are different features of test strategies? Describe any one feature.** 4 M

	Ans	A test strategy is the test approach implementation of a project, defines how testing would be carried out. (1) Analytical Approach: This test approach is based on an analysis of some factor, which strongly effects the testing environment, e.g., Requirements might be analyzed in order to design a test approach such that the most important requirements get tested first and test cases for other requirements get designed / executed later. Another exercise commonly performed is Risk Analysis, where tests are designed and prioritized such that the critical risks get eliminated at the earliest possible time. This approach is an example of preventive test approach, since we are analyzing and prioritizing test cases early on, based on an analysis of some factor contributing to the testing environment. (2) Model-based approach: the tests are designed based on some mathematical or stochastic (statistical) model of the object functionality. As an example, if a model predicts the failure rates of a particular system (or software) under some conditions, and the failure rate of our product is as stipulated by the model under the specified conditions, then our product is assumed to be working fine. This approach pays emphasis on identification and selection of the appropriate model during the early stages of SDLC and is a preventive test approach. (3) Methodical Approach: This test approach depends heavily upon following a pre-determined method to perform testing. The method used can vary widely, ranging from adherence to certain checklists to error guessing and experience-based approaches. Here, tests are designed, executed and implemented in accordance with the selected method. Actual testing effort may get started early on or later during the SDLC when following this testing approach. (4) Process-or Standard-compliant Approach: As the name suggests, this testing approach recommends designing and creation of test assets based on some externally developed industry standard, e.g., Tests could be designed based on IEEE 829 standards. As an alternative, one of the agile methodologies, such as Extreme Programming (XP), might be adopted. Again, actual testing effort might get started early on or later during the SDLC when this approach is selected. (5) Dynamic (Heuristic) Approach: This test approach involves performing heuristic testing. Exploratory testing is a good example of the type of testing performed, when this approach is followed. Here, the tests are designed and executed simultaneously, and hence, it is a reactive test approach. (6) Regression-averse Approach: This approach recommends designing tests such that regression defects get detected at the earliest. This may involve extensive automation of functional regression tests as well as re-use of existing test material.	List of test Strategies/test approaches: 1 mark, explanation of any one: 3 marks (any another relevant answer shall be given marks)
	B	**Attempt any ONE of the following :**	**6 M**
	a	**Explain V-Model with labeled diagram. State its any two advantages and disadvantages.**	6 M
	Ans	V-model means verification and validation model. It is sequential path of execution of processes. Each phase must be completed before the next phase begins. Under V-	Explanation on: 1 mark

model, the corresponding testing phase of the development phase is planned in parallel. So there is verification on one side of V & validation phase on the other side of V.

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 base on

Validation:

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.

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

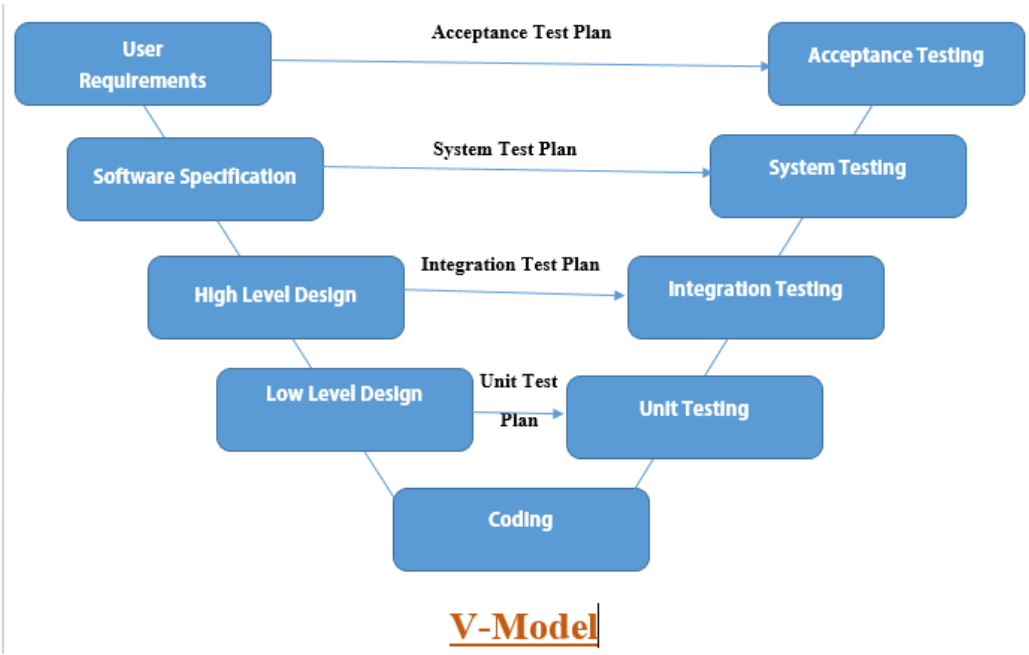
Integration Testing: It is associated with high level design phase & it is performed to test the coexistence & communication of the internal modules within the system

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.

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

and diagram: 1 mark , Advantages:2 marks , Disadvantages: 2 marks (any two)

load & performance defects in the actual user environment.



V-Model

Advantages of V-Model:

- Simple and easy to use.
- Testing activities like planning, test designing happens well before coding.
- Saves a lot of time.
- Higher chance of success over the waterfall model.
- Proactive defect tracking- that is defect is found at early stage.
- Avoids the downward flow of the defects.
- Works well for small projects where requirements are easily understood.

Disadvantages of V-Model:

- Very rigid and least flexible.
- Software is developed during the implementation phase, so no early proto types of the software are produced.
- If any changes happen in mid-way, then the test documents along with requirement documents has to be updated.

	b	**Give any six differences between white box testing and black box testing.**	6 M

	Ans	Comparison **White box testing** / **Black box testing** This needs the knowledge of software in detail. / This does not need the knowledge of software in detail This can be performed by only developers and professional testers. / This can be performed by end users or anyone. The testing is proper here with respect to the domain ,data etc. / The testing is only by trial and error methods. It is suited for algorithm testing. / It is not suited for algorithm testing. It is exhaustive and time consuming. / It is least exhaustive and least time consuming. It is a structural testing of a system. / It is a behavioral testing of a system. It is also called as transparent box or glass box testing. / It is also called as opaque box, dark box testing It includes walkthroughs, inspections, documentation testing etc. / It includes equivalence partitioning, boundary value analysis, etc.	White box testing and black box testing any 6 differences :6 marks, 1 mark each
5		**Attempt any TWO of the following :**	**16 M**
	a	**Explain in detail, how to prepare a test plan with suitable example. Also, explain risk management during test planning.**	8 M
	Ans	The test plan acts as the anchor for the execution, tracking and reporting of the entire testing project and covers. **Preparing test plan:** • What needs to be tested – the scope of testing, including clear identification of what will be the tested & what will not be tested. • 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. • What resources are needed for testing- computer as well as human resources. • The time lines by which the testing activities will be performed. • Risks that may be faced in all of the above, with appropriate mitigation and contingency plans.	Preparation of test plan: 4marks, example: 2marks, Risk management: 2marks Any other relevant test plan format shall be given marks

1. Deciding Test approach/ strategy:

This includes identifying.

- What type of testing would use for testing functionality?
- What are the configurations for testing features?
- What integration testing would you do to ensure these features work together?
- What "non-functional" tests would you need to do?

2. Setting up criteria for testing:

Some of the typical suspension criteria include:

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

3. Identifying responsibilities, staffing & Training needs:

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

4. 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.

5. Identifying Test Deliverables:

It includes:

- Test plan itself, test case design specification, test cases, test logs & test summary report

6. Testing task:

Size and Effort estimation:

- This gives estimation in terms of size, effort & schedule of testing project.

Example:

1. Introduction

   1.1 Scope What features are to be tested and what features will not be tested what combinations of environment are to be tested and what not.

2. References

3. Test Methodology and Strategy/Approach

4. Test Criteria

   4.1 Entry Criteria

   4.2 Exit Criteria

   4.3 Suspension Criteria

   4.4 Resumption Criteria

5. Assumptions, Dependencies, and Risks

   5.1 Assumption

   5.2 Dependencies

   5.3 Risk and Risk Management Plans

6. Estimations

   6.1 Size Estimate

   6.2 Effort Estimate

   6.3 Schedule Estimate

7. Test Deliverables and Milestones

8. Responsibilities

9. Resource Requirement

   9.1 Hardware Resources

   9.2 Software Resources

   9.3 People Resources

		9.4 Other Resources 10. Training Requirements 10.1 Detail of Training Required 10.2 Possible Attendees 10.3 Any Constrains 11. Defect Logging and Tracking Process 12. Metrics Plan 13. Product Release Criteria **Risk management in test planning:** • A common and very useful part of test planning is to identify potential problem or risky areas of the project—ones that could have an impact on the test effort. • Suppose that you and 10 other new testers, whose total software test experience was reading this book, were assigned to test the software for a new nuclear power plant. That would be a risk. • Maybe no one realizes that some new software has to be tested against 1,500 modems and there's no time in the project schedule for it is the another risk. • As a software tester, he will be responsible for identifying risks during the planning process and communicating your concerns to your manager and the project manager. • These risks will be identified in the software test plan and accounted for in the schedule. Some will come true; others will turn out to be benign. The important thing is to identify them early so that they don't appear as a surprise late in the project. • These risks should be identified well ahead of time and the effective management is necessary for them. • Proactive and reactive risks should be studied properly and should be evaluated to make them and their impact low.	
	b	**With the help of suitable example, explain the use of decision table in the black box testing.**	8 M
	Ans	Decision table	Explanation: 3marks, Example: 3marks, Importance:

				2mark

Conditions	TC1	TC2	TC3	TC4
Request login	0	1	1	1
Valid user name entered	X	0	1	1
Valid password entered	X	X	0	1
Actions				
Offer recovery credentials	0	1	1	0
Activate entrybox user name	0	1	1	0
Activate entrybox password	0	0	1	0
Enter priviliged area	0	0	0	1

i. Decision table testing is black box test design technique to determine the test scenarios for complex business logic.

ii. Decision tables provide a systematic way of stating complex business rules, which is useful for developers as well as for testers.

iii. Decision tables can be used in test design whether or not they are used in specifications, as they help testers explore the effects of combinations of different inputs and other software states that must correctly implement business rules.

iv. It helps the developers to do a better job can also lead to better relationships with them.

v. Testing combinations can be a challenge, as the number of combinations can often be huge.

vi. Testing all combinations may be impractical if not impossible.

vii. We have to be satisfied with testing just a small subset of combinations but making the choice of which combinations to test and which to leave out is also important.

viii. If you do not have a systematic way of selecting combinations, an arbitrary subset will be used and this may well result in an ineffective test effort.

Importance of Decision Table: Essentially it is a structured exercise to formulate requirements when dealing with complex business rules. Decision tables are used to model complicated logic. They can make it easy to see that all possible combinations of conditions have been considered and when conditions are missed, it is easy to see.

	c	**Differentiate between alpha testing and beta testing with parameters.**	8 M

	Ans	**Parameters**	**Alpha Testing**	**Beta Testing**	1 mark for each parameter
		Performed	Alpha testing is a type of acceptance testing,	Beta testing is always conducted usually in the	

by	performed within the organization, by the developers within the site of development itself.	real time environment by customers or end users at the clients place (not in the organization).
Performance environment	For alpha testing it requires lab environment or testing environment, and the developer is present as well.	Beta testing doesn't require any specific environment or any independent testing team, it can be done by the client itself.
When it is done?	Alpha testing is done before the software or the product has been launched in the market, this is to make sure that the product has no fault.	Beta testing is usually done at the time of the software product marketing. It is not a complex process.
Techniques required	Alpha testing requires both white box as well as the black box techniques for testing	While beta testing requires only the black box testing techniques or even the functional techniques.
Final outcome	Alpha testing ensures the quality of the product before it goes for beta testing.	Beta testing also works on the quality parameter but it decides whether the product is ready for real time users or consumers.
Open for public	NO	YES
Qualification required	Alpha testing compulsorily requires High hand professionals to handle the complexity of the software.	Beta testing has no compulsions when it comes to qualifications of the beta testers.
Execution period	Long execution cycle may be required.	Few weeks required.
Problem solving	Critical issues or fixes can be addressed by developers immediately in Alpha testing	Most of the issues or feedback is collected from Beta testing will be implemented in future versions of the product

6		**Attempt any FOUR of the following :**	**16 M**
	a	**What are different techniques for finding defects? Explain any one technique with an example.**	4 M
	Ans	**Different techniques for finding defects are:** 1. Static technique 2. Dynamic technique 3. Operational technique **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** **Different techniques to find the defects are:** a) Quick Attacks b) Equivalence and Boundary Conditions c) Common Failure Modes	2marks: listing techniques, 2marks: explaining any one technique

d) State-Transition Diagrams

e) Use Cases

f) Code-Based Coverage Models

g) Regression and High-Volume Test Techniques

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.

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.
- The more your team stretches itself (using a new database, new programming language, new team members, etc.), riskier the project will be—and, at the same time, the less valuable this technique will be.

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.

- 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,
- 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.

e) Use Cases: 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.

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 prefer 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.

- 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.

g) Regression and High-Volume Test Techniques:

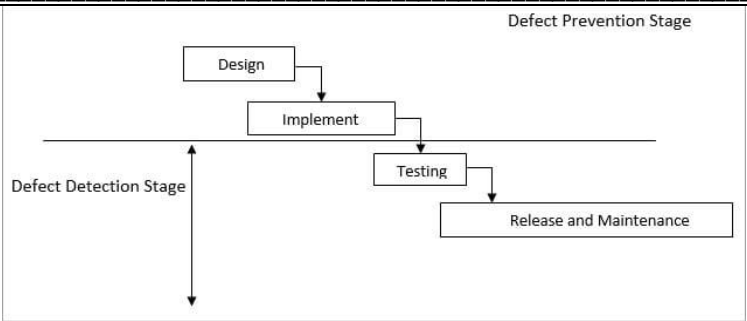
- People spend a lot of money on regression testing, taking the old test ideas described above and rerunning them over and over.
- This is generally done with either expensive users or very expensive programmers spending a lot of time writing and later maintaining those automated tests.

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.

	b	**What is test case? Which parameters are to be considered while documenting test cases?**	4 M
	Ans	**Test Case Specification:** Test case is a well-documented procedure designed to test the functionality of the feature in the system. For designing the test case, it needs to provide set of inputs and its corresponding expected outputs.	Explain test case: 2marks, parameters: 2marks

		Parameters: 1. Test case ID: is the identification number given to each test case. 2. Purpose: defines why the case is being designed. 3. Precondition: for running in the system can be defined, if required, in the test case. 4. Input: should not hypothetical. Actual inputs must be provided, instead of general inputs. Using the test plan as the basis, the testing team designs test case specification, which then becomes the basis for preparing individual test cases. Hence, a test case specification should clearly identify, 1. The purpose of the test: This lists what features or part the test is intended for. 2. Items being tested, along with their version/release numbers as appropriate. 3. Environment that needs to be set up for running the test cases: This includes the hardware environment setup, supporting software environment setup, setup of the product under test. 4. Input data to be used for the test case: The choice of input data will be dependent on the test case itself and the technique followed in the test case. 5. Steps to be followed to execute the test: If automated testing is used, then, these steps ate translated to the scripting language of the tool. 6. The expected results that are considered to be “correct result”. 7. A step to compare the actual result produced with the expected result: This step should do an “intelligent” comparison of the expected and actual results to highlight any discrepancies. 8. Any relationship between this test and other test: These can be in the form of dependencies among the tests or the possibilities of reuse across the tests.	
	c	**Explain defect prevention cycle with neat diagram.**	4 M
	Ans	**Defect Prevention** Defect Prevention is a crucial step or activity in any software development process and as can be seen from the below diagram is pretty much half of our testing tasks:	Explanation: 2marks, Diagram: 2marks **any other diagram can be



considered**

In brief, the following are the defect prevention responsibilities for testers in each of the below stages:

1) Requirement Specification Review:

After understanding customer's requirements prepare your requirement's gist. A review is important at this step- the First level of review should be within the team, followed by another level of external review (by a dev or BA or client) to make sure that all the perspectives are in sync.

2) Design Review:

Design stage can be considered a strategy stage of sorts and going through it will ensure that the QA team understands the pros and cons of each strategy. This kind of critical walkthrough will help unearth any problems with the said strategies and fix them before going further. This can be considered a feasibility study for the strategy (or strategies).

3) Code Review:

There is not a lot for testers to directly get involved in this phase, but the review does go on here too. Developers carry out code inspections, walkthroughs and reviews before they unit and integration test the application.

Defect Prevention Methods and Techniques

Some traditional and common methods that have been in use since a long time for defect prevention are listed below;

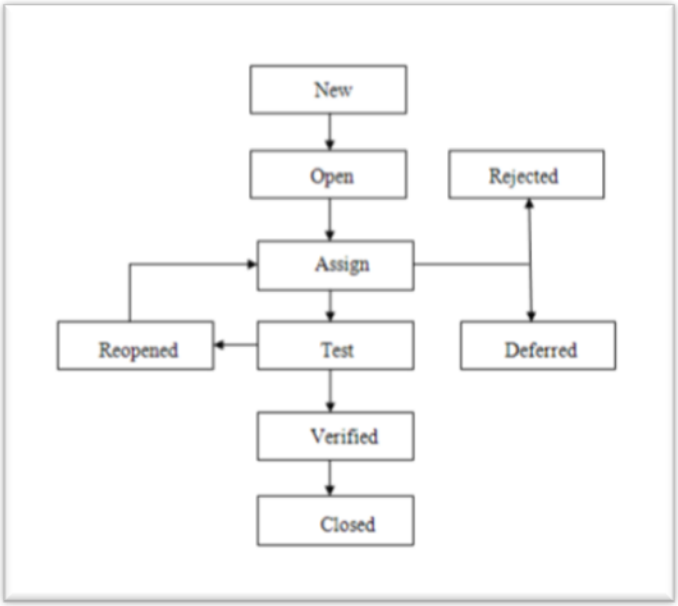
1) Review and Inspection: This method includes the review by an individual team member (self-checking), peer reviews and inspection of all work products.

2) Walkthrough: This is more or less like a review but it's mostly related to comparing the system to the prototype which will give a better idea regarding the correctness and/or the look-and-feel of the system.

3) Defect Logging and Documentation: This method provides some key information, arguments/parameters that can be used to support analyzing defects.

OR

Defect/Bug Life cycle:



1. **New**: When a defect is logged and posted for the first time. Its state is given as new.

2. **Assigned**: After the tester has posted the bug, the lead of the tester approves that the bug is genuine and he assigns the bug to corresponding developer and the developer team. Its state given as assigned.

3. **Open**: At this state the developer has started analyzing and working on the defect fix.

4. **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.

5. **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.

6**. Retest**: 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.

7. **Verified**: The tester tests the bug again after it got fixed by the developer. If the bug is not present in the software, he approves that the bug is fixed and changes the status to "verified".

8. **Reopen**: If the bug still exists even after the bug is fixed by the developer, the tester changes the status to "reopened". The bug goes through the life cycle once again.
9. 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.

10**. 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".

11. **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".

		12. **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. 13. **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.	
	d	**List and explain any two types of standards in test management.**	4 M
	Ans	**Internal standards are:** 1. Naming and storage conventions for test artifacts. 2. Document standards 3. Test coding standards 4. Test reporting standards. **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. **2. Documentation standards:** 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. **3. Test coding standards**:	List: 2marks, Explanation: 2marks

a) Enforce right type of initialization

b) Stipulate ways of naming variables.

c) Encourage reusability of test artifacts

d) Provide standard interfaces to external entities like operating system, hardware and so on.

4. **Test reporting standard**:

All the stakeholders must get a consistent and timely view of the progress of tests. It provides guidelines on the level of details that should be present in the test report, their standard formats and contents.

External Standards:

These are the standards made by an entity external to an organization. These standards are standards that a product should comply with, are externally visible and are usually stipulated by external parties.

The three types of external standards are:

- Customer standard: refer to something defined by the customer as per his /her business requirement for the given product.
- National Standard: refer to something defined by the regulatory entities of the country where the supplier / customer reside.
- International Standard: are defined at international level and these are applicable to all customers across the globe.

	e	**Explain performance testing with an example.**	4 M
	Ans	Performance testing, a non-functional testing technique performed to determine the system parameters in terms of responsiveness and stability under various workload. Performance testing measures the quality attributes of the system, such as scalability, reliability and resource usage. For example, even the fastest computer will function poorly on today's web if the bandwidth is less than 1 megabit per second (Mbps). Slow data transfer rates might be inherent in hardware, but can also result from software-related problems, such as too many applications running at the same time or a corrupted file in a web browser.	Explanation: 3marks, example: 1mark

Performance testing techniques:

- **Load testing -** It is the simplest form of testing conducted to understand the behaviour of the system under a specific load. Load testing will result in measuring important business critical transactions and load on the database, application server, etc., are also monitored.
- **Stress testing -** It is performed to find the upper limit capacity of the system and also to determine how the system performs if the current load goes well above the expected maximum.