MAHARASHTRA STATE BOARD OF TECHNICAL EDUCATION
(Autonomous)
(ISO/IEC - 27001 - 2005 Certified)

WINTER – 2019 EXAMINATION
MODEL ANSWER

Subject: 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)** **Ans.**	**Attempt any FIVE of the following:** **State any two advantages and disadvantages of 'V' model.** The advantage of the V-Model method is that it is very easy to understand and apply. The simplicity of this model also makes it easier to manage. The disadvantage is that the model is not flexible to changes and just in case there is a requirement change, which is very common in today's dynamic world, it becomes very expensive to make the change. **The advantages of the V-Model method are as follows:** • This is a highly-disciplined model and Phases are completed one at a time. • Works well for smaller projects where requirements are very well understood. • Simple and easy to understand and use. • Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.	**5 x 4=20** **4M** ***Any two advantages 1M each***

		The disadvantages of the V-Model method are as follows: • High risk and uncertainty. • Not a good model for complex and object-oriented projects. • Poor model for long and ongoing projects. • Not suitable for the projects where requirements are at a moderate to high risk of changing. • Once an application is in the testing stage, it is difficult to go back and change functionality. • No working software is produced until late during the life cycle.	*Any two disadvantages 1M each*
	(b) Ans.	**Explain the difference between walkthrough and inspection.** **Walkthroughs:** Author presents their developed code to an audience of peers. Peers question and comment on the code to identify as many defects as possible. It involves no prior preparation by the audience. Usually involves minimal documentation of either the process or any arising issues. Defect tracking in walk through is inconsistent. A walk through is an evaluation process which is an informal meeting, which does not require preparation. The product is described by the produced and queries for the comments of participants. The results are the information to the participants about the product instead of correcting it. **Inspection:** is used to verify the compliance of the product with specified standards and requirements. It is done by examining, comparing the product with the designs, code, artefacts and any other documentation available. It needs proper planning and overviews are done on the planning to ensure that inspections are held properly. Lots of preparations are needed, meetings are held to do inspections and then on the basis of the feedback of the inspection, rework is done. Inspection is deserving method with careful consideration of an organization, which concerns about the quality of the product. The process is being done by the quality control department. Inspection is a disciplined practice for correcting defects in software artefacts. **OR**	4M *Any four differences 1M each*

Sr. No.	Walkthrough	Inspection
1	Informal	Formal
2	Initiated by the author	Initiated by the project team

3	Unplanned.	Planned meeting with fixed roles assigned to all the members involved		
4	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.		
5	Author makes a note of defects and suggestions offered by team mate	Recorder records the defects		
6	Informal, so there is no moderator	Moderator has a role in making sure that the discussions proceed on the productive lines		

(c) Ans. **Describe SDLC in software testing.** **4M**

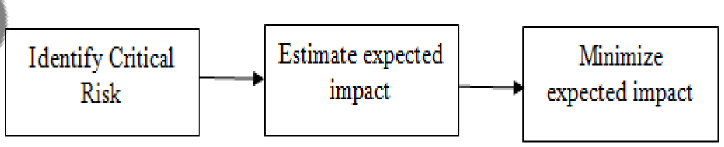
Software Development Life Cycle (SDLC) is a process used by the software industry to design, develop and test high quality software. The SDLC aims to produce a high-quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates. It is also called as Software Development Process. SDLC is a framework defining tasks performed at each step in the software development process. ISO/IEC 12207 is an international standard for software life-cycle processes. It aims to be the standard that defines all the tasks required for developing and maintaining software.

Description 4M

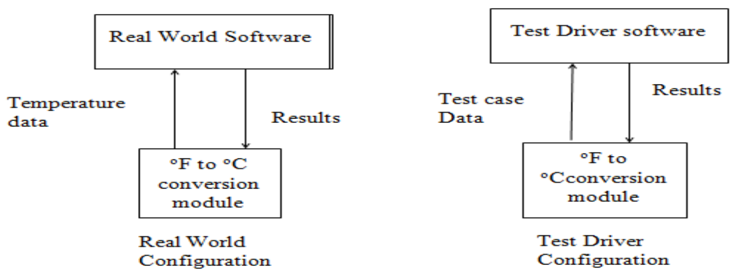
SDLC is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software. The life cycle defines a methodology for improving the quality of software and the overall development process.

Phases of SDLC are as follows:

1) Requirements Gathering and Analysis: During requirement gathering, the specific requirement of the software to be built are gathered and documented. The requirements are documented in the form of SRS document. It acts as bridge between the customer and the organization.

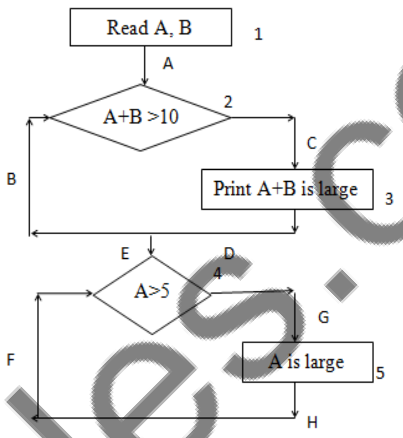
		2) Planning: The purpose of planning phase is to make a schedule, the scope, and resource requirements for a release. A plan explains how the requirements will be met and by which time. At the end of this stage, both project plan and test plan documents are delivered. 3) Design: The purpose of design phase is to figure out how to satisfy the requirements enumerated in SRS. The design phase produces a representation which will be used by the development phase. 4) Development and Coding: The development and coding phase comprises of coding the programs in the chosen programming language. It produces software that meets the requirement and the design. 5) Testing: Testing is process of exercising the software product in predefined ways to check if the behaviour is same as expected behaviour. By testing the product, an organization identifies and removes as many defects as possible before deployment. 6) Deployment and Maintenance: Once a product is tested, it is given to the customers who deploy it in their environments. In maintenance phase wherein the product is maintained or changed to satisfy the changes that arise from customer expectations, environmental changes etc.	
	(d) Ans.	**State how to minimize risk impact while estimating defect.** Minimizing expected impact involves a combination of the following three strategies:  Identify Critical Risk Estimate expected impact Minimize expected impact ● **Eliminate the Risk**: While this is not always possible or desirable, there are situations where the best strategy will be simply to eliminate the risk altogether. For example, reducing the scope of a system, or deciding not to use the latest unproven technology are ways to eliminate certain risks altogether. ● **Reduce the Probability of a Risk Becoming a Problem**: Most strategies will fall into this category. Inspections and testing are	**4M** ***Description 4M***

		examples of approaches that reduce, but do not eliminate, the probability of problems. ● **Reduce the Impact if there is a Problem**: In some situations, the risk cannot be eliminated, and even when the probability of a problem is low, the expected impact is high. In these cases, the best strategy may be to explore ways to reduce the impact if there is a problem. Contingency plans and disaster recovery plans would be examples of this strategy. Once the critical risks are identified, the financial impact of each risk should be estimated. This can be done by assessing the impact, in dollars, if the risk does become a problem combined with the probability that the risk will become a problem. The product of these two numbers is the expected impact of the risk. The expected impact of a risk (E) is calculated as E = P * I, where: P= Probability of risk becoming a problem. I= Impact in dollars if risk becoming a problem.	
	(e) **Ans.**	**What are the benefits of automation?** **Benefits of Automation are as follows:** • **Reliable:** Tests perform precisely the same operations each time they are run, thereby eliminating human error. • **Repeatable:** You can test how the software reacts under repeated execution of the same operations. • **Programmable:** You can program sophisticated tests that bring out hidden information from the application. • **Comprehensive:** You can build a suite of tests that covers every feature in your application. • **Reusable:** You can reuse tests on different versions of application, even if users interface changes. • **Better quality software:** Because you can run more tests in less time with fewer resources. • **Fast:** Automated tools run tests significantly faster than human errors. • **Cost Reduction:** As the number of resources for regression test are reduced.	**4M** ***Any four benefits 1M each***
	(f) **Ans.**	**State the different errors found in static testing.** Static Testing, a software testing technique in which the software is tested without executing the code. It has two parts as listed below: • Review - Typically used to find and eliminate errors or ambiguities in documents such as requirements, design, test cases,	**4M**

		etc. • Static analysis - The code written by developers are analysed (usually by tools) for structural defects that may lead to defects. Following are the types of defects found by the tools during static analysis: • A variable with an undefined value. • Inconsistent interface between modules and components. • Variables that are declared but never used. • Unreachable code (or) Dead Code. • Programming standards violations. • Security vulnerabilities. • Syntax violations.	***Any four errors 1M each***
	(g) **Ans.**	**Explain the concept of stub and driver.** The most common approach to unit testing requires drivers and stubs to be written. Driver and stubs are special purpose arrangements, generally code, required to test units individually which can act as an input to the unit /module and can take output from unit/module. The driver simulates a calling unit and the stub simulates a called unit. A component is not a standalone program; driver and/or stub software must often be developed for each unit test. In most applications a driver is nothing more than a "main program" that accepts test case data, passes such data to the component (to be tested) and prints relevant results. Stubs serve to replace modules that are subordinate (invoked by) the component to be tested. A **stub** or "dummy subprogram" uses the sub – ordinate module's interface, may do minimal data manipulation, prints verification of entry and returns control to the module undergoing testing. Both are the software that must be written but that is not delivered with the final software product. Example:  Real World Software Temperature data Results °F to °C conversion module Real World Configuration Test Driver software Test case Data Results °F to °Cconversion module Test Driver Configuration	**4M** ***Explanation of stub 2M and driver 2M***

		A test **driver** can replace the real software and more efficiently test a low level module. Drivers send test case data to the modules under test, read back the results, and verify that they are correct. A test stub sends test data up to the module being tested.	
2.	**(a)** **Ans.**	**Attempt any FOUR of the following:** **What is meaning of "test to pass" and "test to fail"? Give example.** There are two fundamental approaches to testing software: Test-to-pass and test-to-fail. When you **test-to-pass**, you really assure only that the software minimally works applying the simplest and most straightforward test cases. An example: Press button A, then press button B, then press the Submit button. Testing to pass is typically used when an application or a website is either in its proof of concept stage or is in its infancy and is so fragile that any deviation from controlled steps is likely to produce a fatal error. **Testing to fail** involves testing a feature in every conceivable way possible. Once an application or a website has evolved beyond the initial proof of concept phase, it should be tested to fail, and aggressively. Staying with the above example, a tester might click button A or button B twice before clicking Submit. He may click them out of order, click one or the other several times, or just go right for the Submit button without clicking either of the first two buttons.	**4 x 4=16** **4M** ***Each definition 1M*** ***Each example 1M***
	(b) **Ans.**	**Explain stress testing with reference to "MSBTE" website testing.** • Stress Testing is defined as a type of Software Testing that verified the stability & reliability of the system. This test mainly determines the system on its robustness and error handling under extremely heavy load conditions. • It even tests beyond the normal operating point and evaluates how the system works under those extreme conditions. Stress Testing is done to make sure that the system would not crash under crunch situations. • The goal of stress testing is to analyze the behavior of the system after a failure. For stress testing to be successful, a system should display an appropriate error message while it is under extreme conditions.	**4M** ***Explanation 4M***

		• To conduct Stress Testing, sometimes, massive data sets may be used which may get lost during Stress Testing. Testers should not lose this security-related data while doing stress testing. • The main purpose of stress testing is to make sure that the system recovers after failure which is called as **recoverability**. • For Example, In stress testing of MSBTE online result display, the resources used will be less than the requirement. For e.g. Provide less RAM for the server, or decrease the bandwidth of the internet connection, or provide less hits for page. If the system has limited resources available, the response of the online result system may deteriorate due to non-availability of the resources. It tries to break the page, site or connection under test by overwhelming its resources in order to find the circumstances under which it will crash. It is also a type of load testing. It is designed to determine the behaviour of the software under abnormal situations. In stress testing test cases are designed to execute the system in such a way that abnormal conditions.	
	(c) **Ans.**	**Explain branch coverage with proper example.** **Branch Coverage:** Attempting to cover all paths in the software is called path testing. The simplest form of path testing is called branch coverage testing. The goal of branch coverage is to ensure that whenever a program can jump, it jumps to all possible destinations. Branch coverage is a testing method, which aims to ensure that each one of the possible branch from each decision point is executed at least once and thereby ensuring that all reachable code is executed. Branch coverage is also known as Decision Coverage. Formula: $$\text{Branch Coverage} = \frac{\text{Number of decisions outcomes tested X 100}}{\text{Total Number of decision Outcomes}}$$ ***Example:*** Read A Read B If A+B > 10 then Print “A+B is Large” End If If A>5 then	**4M** ***Explanation 2M*** ***Example 2M***

		Print "A Large" End if To calculate Branch Coverage, one has to find out the minimum number of paths which will ensure that all the edges are covered. In the above example there is no single path which will ensure coverage of all the edges at once. The aim is cover all possible true /false decisions. 1. 1A-2C-3D-E-4G-5H 2. 1A-2B-E-4F Hence Branch Coverage is 2.  Read A, B 1 A A+B >10 2 C B Print A+B is large 3 E D 4 A>5 G F A is large 5 H	
	(d) **Ans.**	**List any four advantages of acceptance test before launching of any software.** **Advantages:** 1. This testing gives user an opportunity to ensure that software meets user requirements, before actually accepting it from the developer. 2. It is easier and simpler to run an acceptance test compared to another types of test. 3. It enables both users and software developers to identify and resolve problems in software. 4. This testing determines the readiness of software to perform operations. 5. It decreases the possibility of software failure to a large extent.	**4M** ***Any four advantages 1M each***
	(e) **Ans.**	**Explain test deliverables in detail.** • Test deliverables identifies the deliverable documents from the test process. Test input and output data should be identified as deliverables. • Testing report logs, test incident reports, test summary reports and metrics reports must be considered testing deliverables. • The deliverable include following: 1. Test plan document 2. Test cases 3. Test design specifications	**4M** ***Explanation 4M***

		4. Tools and their outputs 5. Simulators 6. Static and dynamic generators. 7. Error logs and execution logs. 8. Problem reports and corrective actions. 9. Test summary report.	
	(f) **Ans.**	**Explain different techniques to find defects.** • Whenever, a software product is examined, different types of defects or bugs get encountered in software. • Defects are found either by pre planned activities specifically intended to uncover defects or by accident. • Techniques to find defects can be divided into following three categories: 1. Static Techniques: Testing that is done without physically executing a program or system. A code review, walkthrough, inspections etc. are the examples of static testing technique. 2. Dynamic Techniques: Testing in which system components are physically executed to identify defects. Execution of test cases is an example of a dynamic testing technique. 3. Operational Techniques: An operational system produces a deliverable containing a defect found by users, customers, or control personnel i.e., the defect is found as a result of a failure.	**4M** ***Explanation 4M***
3.	**(a)** **Ans.**	**Attempt any FOUR of the following:** **Differentiate between GUI and Usability testing.**	**4 x 4=16** **4M** ***Any four differences 1M each***

GUI Testing	**Usability testing**
1. In GUI Testing tester tests the application front end design to see whether its meets the client requirements or not.	1. In Usability Testing tester tests that whether the application is user friendly or not by checking how easily user can access the application.
2. In GUI Testing we check whether the design and layout of application as per the standards and client requirements or not.	2. In Usability Testing we check whether the design and layout of application is easy to use or not means it is user friendly or not.
3. GUI Testing is more concerned with look and feel of the application means how	3. Usability Testing is more concerned with easiness and user friendliness of the

		people react and feel after look in to the application so its testing is done accordingly that.	application means how people react after using the application means application is easy to use or not so it's testing is done accordingly that.	
		4. In GUI Testing tester tests the appearance of the software.	4. In Usability Testing tester tests the easiness to use the software.	
		5. GUI Testing is done to ensure it meets the design specifications like links, colors, fonts, font sizes, fields etc are displayed as specified in SRS or as specified in client requirements.	5. Usability Testing is done to ensure that the GUI is well designed and easy to use like links and buttons are easily clickable and leaving any of the mandatory field blank gives the proper message that please enter the xyz in mandatory field.	
		6. GUI Testing is done by keeping in mind the look and feel of application means how application looks.	6. Usability Testing is done by keeping the end user in mind.	
		7. It stands for Graphical User Interface. It is nothing its only confirm the design specifications with the application.	7. It is done to ensure that the GUI is well designed and easy to use.	
		8. It is done on different platforms to verify the Look and Feel Testing. (Look and Feel of the application).	8. It is done to verify how much the application is user friendly to an end user.	
		9. In GUI Testing, tester test whether the front end design of the system is meeting with project standards or not.	9. In Usability Testing, tester tests whether the control flow of the system is convenient for end user or not.	
		10. In this testing we just test the appearance of the application.	10. In this testing we test the interaction of functionality with the user is effective or not.	
		11. Example: Example includes colors, fonts, font sizes, buttons, links, icons, placement of data	11. Example: Example includes firstly displayed all mandatory fields, cursor positioning for	

labels and fields etc. are displayed as specified or not.	enter the data into the right field, tab button should work easily etc.
12. In GUI Testing we only focus on the interface of the application.	12. Quality of product is depending on Usability Testing.
13. In this testing we test only the front end of the application.	13. In this Testing we test the overall working of application according to a non-technical user's point of view.

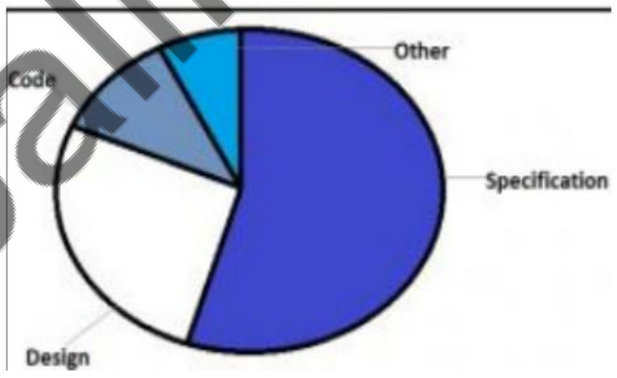
(b) **Explain boundary condition and sub-boundary condition with example.** **4M**

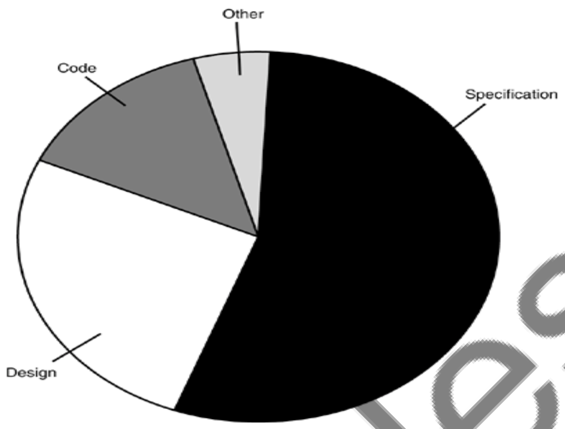
Ans.

Boundary condition with example 2M

- Most of the defects in software products hover around conditions and boundaries. By conditions, we mean situations wherein, based on the values of various variables, certain actions would have to be taken. 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 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 centre 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.
- Same examples of Boundary value analysis concept are:
- 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.
- For Example: A system can accept the numbers from 1 to 10 numeric values. All other numbers are invalid values. Under this

		technique, boundary values 0, 1, 2, 9, 10, 11 can be tested.
		• Another Example is in exam has a pass boundary at 40 percent, merit at 75 percent and distinction at 85 percent. The Valid Boundary values for this scenario will be as follows:
		49, 50 - for pass
		74, 75 - for merit
		84, 85 - for distinction
		• Boundary values are validated against both the valid boundaries and invalid boundaries. The Invalid Boundary Cases for the above example can be given as follows 0 - for lower limit boundary value 101 - for upper limit boundary value Boundary value analysis is a black box testing and is also applies to white box testing. Internal data structures like arrays, stacks and queues need to be checked for boundary or limit conditions; when there are linked lists used as internal structures, the behavior of the list at the beginning and end have to be tested thoroughly.
		• Boundary value analysis help identify the test cases that are most likely to uncover defects.
		
		Sub-Boundary Conditions
		1. They're the ones defined in the specification or evident when using the software.
		2. Some boundaries, though, that are internal to the software aren't necessarily apparent to an end user but still need to be checked by the software tester.
		3. These are known as sub-boundary conditions or internal boundary conditions.
		4. In the given example the sub boundary condition is the value of factorial
		For example
		#include
		void main()
		{
		int i , fact=1, n;
		printf("enter the number ");
		scanf("%d",&n);
		for(i =1 ;i <=n;i++)
		fact = fact * i; "%d", fact);
		printf ("the factorial of a number is "%d", fact);
			Sub boundary condition with example 2M

		} The boundary condition in the above example is for the integer variable.	
	(c) Ans.	**What are different causes of software defects?** Causes of Software Defects: Different causes of software defects are as given below: In software defects occur due to various reasons. 1. One of the extreme causes is the specification. 2. Specifications are the largest producer of defects. 3. Either specifications are not written, specifications are not thorough enough, constantly changing or not communicated well to the development team. 4. Another bigger reason is that software is always created by human beings. 5. They know numerous things but are not expert and might make mistakes. 6. Further, there are deadlines to deliver the project on time. So increasing pressure and workload conduct in no time to check, compromise on quality and incomplete systems. So this leads to occurrence of defects in softwares. Following diagram depicts the causes of defects in softwares:  **OR**	**4M** ***Any four causes 1M each***



	(d) Ans.	**Explain equivalence partitioning with example.** 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.	**4M** *Explanation 2M*
		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**.	***Example 2M***

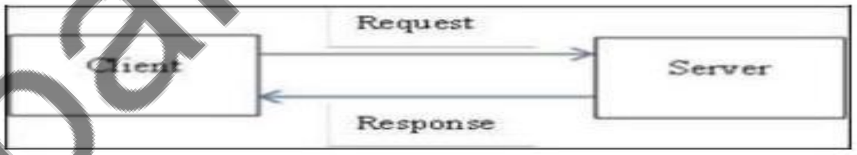
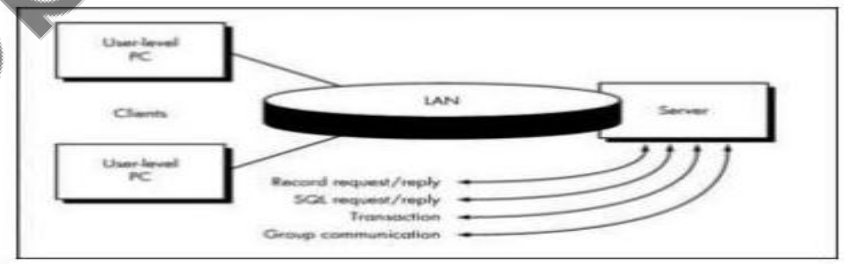
Age group	Extra Premium
Under 35	Rs.1500
35-59	Rs. 2500
60+	Rs. 4000

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

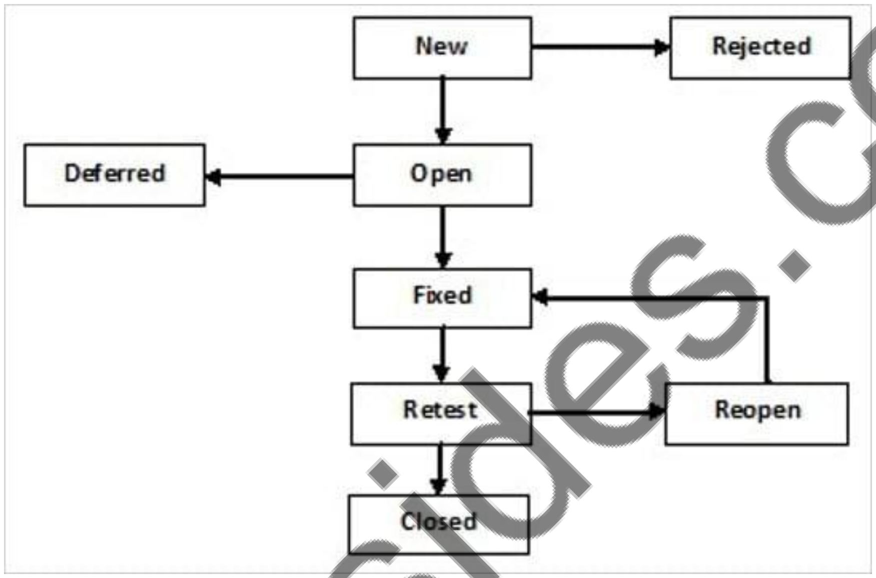
- Below 35 years of age (valid input)

		• Between 35 and 59 years of age (valid input) • Above 6 years of age (valid input) • Negative age (invalid input) • Age as 0(invalid input) Age as any three-digit number(valid input)	
	(e) Ans.	**What are the objectives of software testing?** **Objectives of Testing:** 1. Finding defects which may get created by the programmer while developing the software. 2. Gaining confidence in and providing information about the level of quality. 3. To prevent defects. 4. To make sure that the end result meets the business and user requirements. 5. To ensure that it satisfies the BRS that is Business Requirement Specification and SRS that is System Requirement Specifications. 6. To gain the confidence of the customers by providing them a quality product.	**4M** ***Any four objectives 1M each***
	(f) Ans.	**What are metrics? Explain any one detail.** **Metrics:** 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. Types of Metrics: • Process quality • Product quality • Objective Metrics • Subjective Metrics **Process quality:** Activities related to the production of software, tasks or milestones. 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. 4. The only way to know how/where to improve any process is to: • Measure specific attributes of the process. • Develop a set of meaningful metrics based on these attributes.	**4M** ***Metrics definition 2M*** ***Explanation of any one 2M***

		• Use the metrics to provide indicators that will lead to a strategy for improvement. **Product quality:** Explicit result of the software development activity, deliverables, products. 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 software. 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. **Objective Metrics:** 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. **Subjective Metrics:** 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.	
4.	**(a)** **Ans.**	**Attempt any TWO of the following:** **What is test plan? What is its need? List test planning activities.** **Test Plan** A test plan is a systematic approach to testing a system i.e. software. The plan typically contains a detailed understanding of what the eventual testing workflow will be. **Need of test plan:** • Test Plan Ensures all Functional and Design Requirements are implemented as specified in the documentation.	**2 x 8=16** **8M** ***What is test 2M*** ***Need 2M***

		• To provide a procedure for Unit and System Testing. • To identify the documentation process for Unit and System Testing. • To identify the test methods for Unit and System Testing. **Activities** 1. Preparing test plan 2. Scope management 3. Deciding Test approach/ strategy 4. Setting up criteria for testing 5. Identifying responsibilities, staffing & Training needs: 6. Identifying Resource Requirement 7. Identifying Test Deliverables 8. Testing task	***List of activities 4M***
	(b) **Ans.**	**Explain client server testing with suitable diagram.** **Client Server Testing:** 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:   User-level PC Clients LAN Server User-level PC Record request/reply SQL request/reply Transaction Group communication 1. **Component Testing:** One need to define the approach and test plan for testing client and server individually. When server is tested	**8M** ***Explanation 6M*** ***Diagram 2M***

		there is need of a client simulator, 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. **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.	
	(c) **Ans.**	**Explain defect life cycle diagram and different states. Mention defect report template.**	**8M**

		 New Rejected Deferred Open Fixed Retest Reopen Closed 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. **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.	***Defect life cycle diagram 2M*** ***Description 3M***

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.
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.

Defect report template 3M

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.
Defect Priority	Priority of the Defect.
Reported By	The name of the person who reported the defect
Assigned To	The name of the person that is assigned to analyze/fix the defect.
Status	The status of the defect.

5. (A) **Attempt any TWO of the following:** **2 x 6=12**

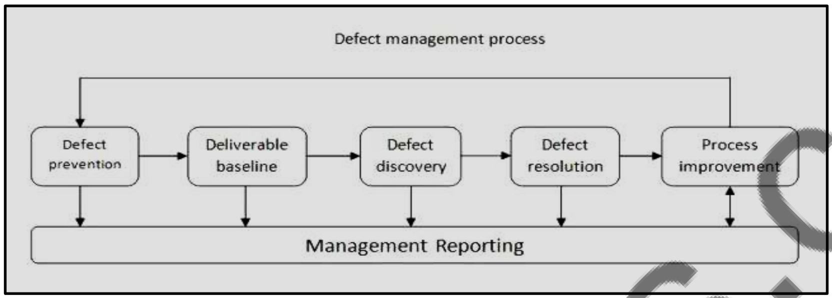
(a) **Design test cases for ATM card operations. (Any six)** **6M**

Ans.

Test case ID	Test case objective	Input data	Expected result	Actual result	Status
TC1	Pin number	valid 4 digits pin	It should accept the valid pin	It accepted the pin	Pass

TC2	Withdrawl	Valid numeric amount	It should accept the valid amount	It accepted the amount	Pass
TC3	Withdrawl	Click on the withdrawl button	It should ask for the amount	It displayed the message as enter the amount	Pass
TC4	Mini statement	Click on mini statement	It should issue the receipt of last 3 transactions	It issued the receipt of last 3 transactions	Pass
TC5	Pin change	Click on pin change button	It should display the message	It displayed the message as Enter the new pin	Pass
TC6	Cancel button	Press the cancel button	it should cancel the current transaction	It cancelled the current transaction	Pass
TC7	Clear button	Press the clear button	it should clear the contents on screen	It cleared the contents of screen	Pass

Any six cases 1M each

	(b) **Ans.**	**Explain defect management process with diagram.**  Defect management process Defect prevention Deliverable baseline Defect discovery Defect resolution Process improvement Management Reporting • Defect Management process must include the appraisal of a defect finding process, software development process and the actions initiated to close the defects and strengthen the product/process associated with development, so that defects are not repeated again and again. It typically includes correction, corrective action and preventive action. It includes, Defect Naming Defect Resolution Once the developer have acknowledged a valid defect , the resolution process starts: • To report on the status of individual defects • To provide tactical information and metrics to help project management, redesign of error prone modules. • To provide strategic information and metrics to senior management, defect trends, problem systems. Process to be improved to either prevents defects or minimizing their impact. To provide insight into the likelihood that target dates and cost estimates will be i. Defect Prevention -- Implementation of techniques, methodology and standard processes to reduce the risk of defects. ii. Deliverable Baseline -- Establishment of milestones where deliverables will be considered complete and ready for further development work. When a deliverable is baseline, any further changes are controlled. Errors in a deliverable are not considered defects until after the deliverable is baseline. iii. Defect Discovery -- Identification and reporting of defects for development team acknowledgment. A defect is only termed discovered when it has been documented and acknowledged as a valid defect by the development team member(s) responsible for the component(s) in error. iv. Defect Resolution -- Work by the development team to prioritize,	**6M** ***Defect management process Diagram 2M*** ***Explanation 4M***

schedule and fix a defect, and document the resolution. This also includes notification back to the tester to ensure that the resolution is verified.

v. Process Improvement -- Identification and analysis of the process in which a defect originated to identify ways to improve the process to prevent future occurrences of similar defects. Also the validation process that should have identified the defect earlier is analyzed to determine ways to strengthen that process.

vi. Management Reporting -- Analysis and reporting of defect information to assist management with risk management, process improvement and project management.

(c) Ans. **Differentiate between black box and white box testing.** **6M**

Any six points 1M each

Comparison Base	Black box testing	White box testing
Other terms	Black box testing is also called data-driven testing, box testing, of functional testing.	White box testing is also called structural testing. Some developers call it clear box testing, code-based testing, glass box testing or transparent box testing.
Meaning	It is a testing approach which is used for testing the software without the knowledge of the internal design or structure of program or application.	It is a testing approach in which internal structure or design is known to the software tester who is going to test the software.
Testing Techniques	Equivalence Partitioning Boundary Value Analysis Decision Table State Transition	Statement Coverage Branch Coverage Path Coverage
Implementation Knowledge	Implementation knowledge is not vital to perform Black Box Testing.	Implementation Knowledge is vital to perform White Box Testing.

Programming Knowledge	It is not required to carry out Black Box Testing.	It is required to carry out White Box Testing.
Prime Focus	Primarily focus on the functionality of the systems under test.	Primarily focus on the testing of the program code of the system under test like branches, code structure, loops, conditions, etc.
Time-period	It is less exhaustive and time-consuming	Exhaustive and time-taking method.
Target	The main aim is to check on what functionally is performing by the system under test.	The main aim of is to check on how the system is performing.
Types of Testing	A. Functional Testing B. Non-functional Testing C. Regression Testing	A. Path Testing B. Loop Testing C. Condition Testing

5. (B) **Attempt any ONE of the following:** **1 x 4=4**

(a) **Write test cases to test "copy and Paste" operation in MS-Paint.** **4M**

Ans. **Test cases for paint Copy , Paste operations:**

Test Case ID	Category	Feature Description	Prerequisite	Test Description	Input Data	Expected Result
PNT_P1.2.3	FUN	Verify the functionality of "Copy"	User is in Edit tab Window.	1. Draw the image in window. 2. Under the Tools tab clicks the Rectangular Marquee & selects the needed area in that image. 3. Click on the Copy option.	Selected input or image component	Which one user should select & copied it should be appearing in the frame on bottom left side.

2M each for correct test case

PNT_ P1.2.4	FUN	Verify the functionality of "Paste".	User is in Edit tab Window.	1. Draw the image in window. 2. Click on the Copy option. 3. Click on the Paste button.	Marked data or image component	Whatever user copy, it should be appear in the window.

(b) Ans. **Describe user acceptance testing.** **4M**

- **User Acceptance** is defined as a type of testing performed by the Client to certify the system with respect to the requirements that was agreed upon. This testing happens in the final phase of testing before moving the software application to the Market or Production environment.
- The main purpose of this testing is to validate the end to end business flow. It does NOT focus on cosmetic errors, Spelling mistakes or System testing. This testing is carried out in a separate testing environment with production like data setup. It is a kind of black box testing where two or more end users will be involved.
- Developers code software based on requirements document which is their "own" understanding of the requirements and may not actually be what the client needs from the software.
- Requirements changes during the course of the project may not be communicated effectively to the developers.
- Acceptance Testing and V-Model
- In V-Model, User acceptance testing corresponds to the requirement phase of the Software Development life cycle (SDLC).

4M for proper description of concept

Step 1) Analysis of Business Requirements
These test scenarios are derived from the following documents:
Project Charter, Business Use Cases, Process Flow Diagrams
Business Requirements Document(BRD), System Requirements Specification(SRS)
Step 2) Creation of UAT Plan:
The UAT test plan outlines the strategy that will be used to verify and ensure an application meets its business requirements. Test scenarios and test cases approach and timelines of testing.
Step 3) Identify Test Scenarios and Test Cases:
Identify the test scenarios with respect to high-level business process

		and create test cases with clear test steps. **Step 4) Preparation of Test Data:** It is best advised to use live data for UAT. Data should be scrambled for privacy and security reasons. Tester should be familiar with the database flow. **Step 5) Run and record the results:** Execute test cases and report bugs if any. Re-test bugs once fixed. Test Management tools can be used for execution. **Step 6) Confirm Business Objectives met:** **Exit criteria for UAT:** Before moving into production, following needs to be considered: No critical defects open Business process works satisfactorily UAT Sign off meeting with all stakeholders **Advantages of acceptance test before launching any software:** 1. Acceptance testing is phase after system testing that is normally done by the customer or representatives of the customer. Due to that customer themselves to quickly judge the qualityof the product. 2. Determine whether the software is fit for the user. 3. Making users confident about product. 4. Determine whether a software system satisfies its acceptance criteria. 5. Enables the buyer to determine whether to accept the system or not.	
6.	**(a)** **Ans.**	**Attempt any FOUR of the following:** **Enlist and describe skills of software tester.** **Skills of software tester are as follows :** **1.Analytical and logical thinking** **i)** The major objective of testing is to identify the hidden errors, not simply prove that the software works. **ii)**For a tester to be effective in his role, he must be able to analyze the given business situation and judge all the possible scenarios. **2.The ability to envision business situations** **i).**A tester should be able to envisage real-time business situations through mental mapping, abstracting the idea inferred from the	**4 x 4=16** **4M** ***1M each for skill explanation (any four)***

specifications.

ii). The real-time business scenarios should crystallize in a tester's mind, and he should think about what the case is rather than what ought to be the case or what he believes the case is.

iii).A tester should be able to anticipate complex problems, in addition to visualizing and articulating them.

3.A sense of intellectual curiosity and creativity

i).A tester should understand that being an intellectual and being intellectually curious are not the same.

ii).A tester should arguably be the latter one -- intellectually curious -which is all about asking questions and not about having answers.

4. A "global" approach

i). Software systems have become extremely complex.

ii). Most of the time, the system designed involves multiple stakeholders, and dealing with such systems is not always easy.

iii).A tester should be able to deal effectively with business situations marked by complexity and the number of interactions with third-party systems.

5.Critical thought and rational enquiry

i). The quality of life of an individual and the quality of what he produces/delivers depends largely on the quality of his thought process.

ii). The thought process of a tester should be undistorted, impartial and without any prejudices.

iii).A tester should be able to take charge of the inherent structures and impose intellectual standards upon the software under test.

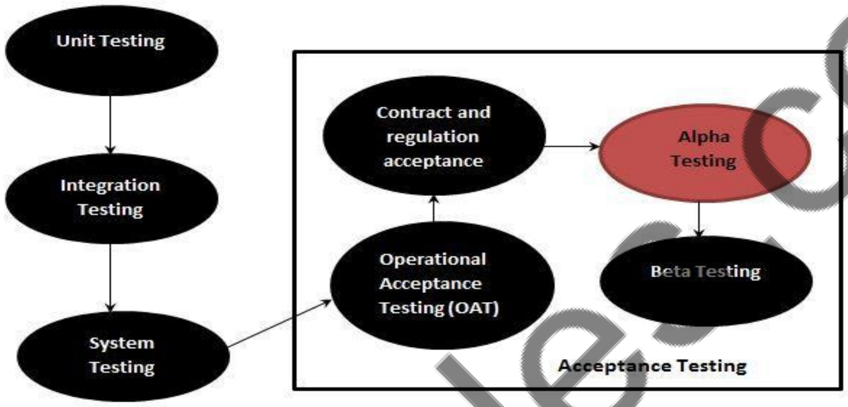
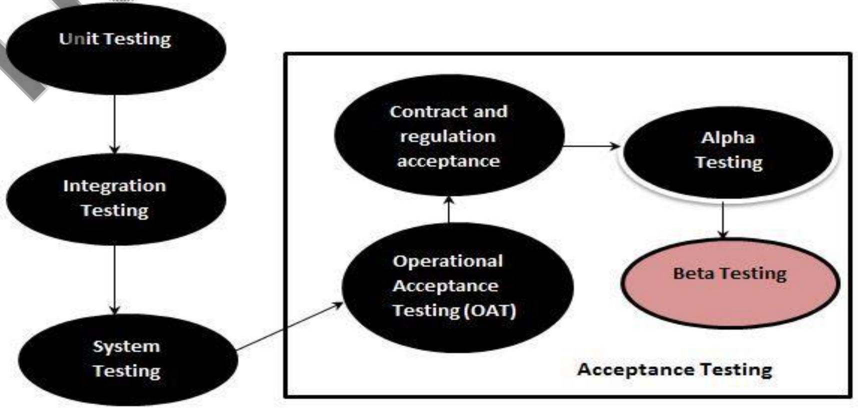
iv). He should be able to raise vital questions precisely and clearly, gather and assess relevant information, interpret it effectively in order to come to well-reasoned conclusions and solutions, and test those conclusions against the given criteria and standards.

6. The ability to apply basic and fundamental knowledge

i). Knowledge in the context of testing can be attributed as the fluid mix of experience, values, contextual information and expert insight.

ii). Those things provide a framework for evaluating the system under test. One can attain knowledge by so many means, but that knowledge is worthwhile only when it adds value to situations encountered.

		iii).A smart tester should be able to apply the knowledge attained over years of experience with the domain, process, product, customers, mistakes and successes in his testing. **7. Continue to learn** **i).**Organizations and business environments change rapidly, which means the approaches and processes that work well today will be outdated tomorrow. **ii).**Therefore, it is imperative that a tester place priority on noticing, adapting and learning from change that is happening around him. **8. Respect for truth and intellectual integrity** **i).**A tester should be able to examine the piece of software under test and the resulting processes, with focus on the given specification, and understand the behavior of the software. **ii).**Being human, a tester may have severe biases, prejudices and intolerances that prevent him from performing well. **9.Planning, time management skills** **i).**Planning is nothing but writing the story of the future. **ii).**A tester needs to have a thorough plan and must develop a well-thought test strategy/approach. **iii).**And that plan must be in place before work begins on any software testing assignment. **iv).**It should describe the items and features to be tested, the test strategy and levels of testing pass/fail criteria, suspension/resumption criteria, schedule, etc. **10. Effective communication skills** **i).**A tester must be able to communicate his thoughts and ideas effectively, using a variety of tools and media. **ii).**He needs to develop and use this skill throughout his career and should learn to communicate effectively to the stakeholders so as to avoid ambiguities and inconsistencies. **iii).**For example, printed presentations should be concise and to the point and should follow logically.	
	(b) **Ans.**	**Describe Alpha and Beta Testing.** **Alpha testing** takes place at the developer's site by the internal teams, before release to external customers. This testing is performed without the involvement of the development teams. i. Alpha Testing - In SDLC	**4M**

		The following diagram explains the fitment of Alpha testing in the software development life cycle.  Unit Testing Integration Testing System Testing Contract and regulation acceptance Alpha Testing Operational Acceptance Testing (OAT) Beta Testing Acceptance Testing **Beta Testing:** i. Beta testing also known as user testing takes place at the end users site by the end users to validate the usability, functionality, compatibility, and reliability testing. ii. Beta testing adds value to the software development life cycle as it allows the "real" customer an opportunity to provide inputs into the design, functionality, and usability of a product. These inputs are not only critical to the success of the product but also an investment into future products when the gathered data is managed effectively. Beta Testing - In SDLC The following diagram explains the fitment of Beta testing in the software development life cycle:  Unit Testing Integration Testing System Testing Contract and regulation acceptance Alpha Testing Operational Acceptance Testing (OAT) Beta Testing Acceptance Testing	***Description of alpha testing 2M & Beta testing 2M***

	(c) **Ans.**	**Explain Regression Testing.** Regression testing a black box testing technique that consists of re-executing those tests that are impacted by the code changes. These tests should be executed as often as possible throughout the software development life cycle. It is performed to validate the build that hasn't changed for a period of time. This build is deployed or shipped to customers. 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. It finds other related bugs. It tests to check the effect on other parts of the program. Regression testing produces Quality software. Validate the parts of software where changes occur. It validates parts of software which may be affected by some changes but otherwise unrelated. It ensures proper functioning of the software, as it was before changes occurred. It enhances quality of software, as it reduces the high risk bugs.	**4M** ***Regression Testing 2M*** ***2 Reason why regression testing is done 2M***
	(d) **Ans.**	**What are the limitations of manual testing?** **Limitations Of Manual Testing:** • Manual testing is not reliable. Using this method test execution is not accurate all the time. • To execute the test cases first time using manual testing will be very much useful. But it is not sure that it will catch the regression defects under frequently changing requirements. • Manual testing will be useful when the test case only needs to run once or twice. • To execute the test cases every time tester requires the same amount of time. • Using manual testing, testing on different machine with different OS platform combination is not possible, concurrently. To execute such task different testers are required. • It does not involve in programming task to fetch hidden information. • Manual testing is slower than automation. Running tests manually can be very time consuming. ✓ Time consuming ✓ Limited support for regression testing ✓ Error prone testing ✓ Impractical performance testing ✓ Non consistent or repeatable ✓ Limited scope	**4M** ***1M for each limitations***

		✓ No batch testing ✓ Need of training ✓ Difficult to manage **OR** **Limitations of Manual Testing are as given below:** **i.** Manual testing is slow and costly. **ii.** It is very labor intensive; it takes a long time to complete tests. **iii.** Manual tests don't scale well. As the complexity of the software increases the complexity of the testing problem grows exponentially. This leads to an increase in total time devoted to testing as well as total cost of testing. **iv.** Manual testing is not consistent or repeatable. Variations in how the tests are performed as inevitable, for various reasons. One tester may approach and perform a certain test differently from another, resulting in different results on the same test, because the tests are not being performed identically. **v.** Lack of training is the common problem, although not unique to manual software testing. **vi.** UI objects size difference and color combinations are not easy to find in manual testing. **vii.** Not suitable for large scale projects and time bound projects. **viii.** Batch testing is not possible, for each and every test execution Human user interaction is mandatory. **ix.** Comparing large amount of data is impractical. Processing change requests during software maintenance takes more time.	
	(e) **Ans.**	**Enlist errors that are uncovered during black box testing.** Black Box Testing uncovered errors: • Incorrect or missing functions. • Interface errors. • Errors in data structures or external database access. • Behavior or performance errors. • Initialization and termination errors. • Logic errors are not done by black-Box testing • Equivalence Partitioning errors • Boundary Value errors • Decision Table Testing parameters • State Transition Testing errors • Comparison Testing parameter errors	**4M** ***Any four error 1M each***

	(f) Ans.	**Explain static and dynamic testing tools.** **Benefits of using testing tools:** • Save Time • Speed • Repeatability • Maintenance of the test suite • Reusable • Increase Coverage • Cost Reduction **Test tools types are:** 1. **Static Testing tools:** Generally used by developers as part of development and component testing process, here code is not executed or run bur the tool itself is executed and source code we are interested in is the input data to the tool. **These are extension of** compiler technology, other than software code , static analysis can also be carried out on things like, static analysis of requirements or static analysis of websites the developer to understand the structure of the code and helpful to enforce coding standards . Static analysis tools for code can help **Features / Characteristics of static analysis tools are:** • To calculate metrics, Cyclomatic complexity or nesting levels. • To enforce coding standards • To analyze structures and dependencies • Help in code understanding • To identify anomalies or defects in the code. **2. Dynamic Testing tools:** They require the code to be in running state They analyze rather than testing. **Features / Characteristics of dynamic analysis tools are:** • To detect memory leaks • To identify pointer arithmetic errors such as null pointers • To identify time dependencies. **Examples of Dynamic testing tools available:** a) Test Driver b) Test Beds c) Emulators d) Mutation analyzers	**4M** ***Static testing tools 2M*** ***Dynamic testing tools 2M***