Basics of Software Testing and Testing Methods

Q1) **What is static testing? State advantages and disadvantages of static testing two each.**

Ans: Static testing is a type testing which requires only source code of the product, not the binaries or executable. Static testing does not involve executing the program on computers but involves select people going through the code to find out whether

1. The code works according functional requirement.
2. The code has been written in accordance with the design developed earlier in the projectlife cycle.
3. The code for any functionality has been missed out.
4. The code handles errors properly.

Advantages:

1. Static testing start early in the life cycle so early feedback on quality issues can be established.

2. As the defects are getting detected at early stage so the rework cost most often relatively low.
3. Development productivity is likely to increase because of the less rework effort.
4. It helps to produce a better product.

Disadvantages:

1. It is time-consuming.
2. The logistics and scheduling can become an issue since multiple peoples are involved.
3. It is not always possible to go through every line of code.
4. Required High- skills.

Q **What is the difference between static & dynamic testing tool?**

Ans:

Static testing tool	Dynamic testing tool
These tools are used by developers as part of the development and component testing process	These tools require the code to be in a "running state"
code is not executed or run but tool itself is executed	They analyse rather than testing
It is extension of compiler technology	They also help to understand sses
It also perform static analysis of requirement or analysis of website	These tool used by developed in component integration testing,, middle ware , testing robustness and security.
Helps to understand the structure of the code and can also be useful to enforce coding standards.	Also performs web site testing to check whether each link does actually link to something else, it can find dead links .

Features /characteristics of static testing tools are: • Checks cyclomatic complexity • Enforces coding standards • Analyse structures and dependencies • Helpful in understanding coding • Identify defects in code.	Features/characteristics of static testing tools are: • Detect memory leak • Identify pointer arithmetic errors , null pointer • Identify time dependence.
Examples. Flow analyzer, path tests, coverage analyzers, Interface analyzers	Examples. Test driver, Test beds, Emulators, Mutation analyzers

Q What is white box testing? Classify static white box testing. State any one situation where white box testing used.

Ans: White box testing:

This is also known as glass box, clear box, and open box testing. In white box testing, test cases are created by looking at the code to detect any potential failure scenarios. The suitable input data for testing various APIs and the special code paths that need to be tested by analysing the source code for the application block. Therefore, the test plans need to be updated before starting white box testing and only after a stable build of the code is available. White box testing assumes that the tester can take a look at the code for the application block and create test cases that look for any potential failure scenarios. During white box testing, analyze the code of the application block and prepare test cases for testing the functionality to ensure that the class is behaving in accordance with the specifications and testing for robustness. A failure of a white box test may result in a change that requires all black box testing to be repeated and white box testing paths to be reviewed and possibly changed.

Classification of white box testing:

- Static Testing- Inspections, Structured Walkthroughs, Technical Review.
- Structural Testing-Code Functional Testing, Code Coverage Testing, Code Complexity Testing.

A simple website application as an example to explain the white box testing. . The end user will simply access the website, Login &Logout; this is very simple and day 2 days life example. As end users point of view users able to access the website from GUI, but inside there are lots of things going on to check the internal things are going right or not, the white box testing method is used. To explain this we have to divide this part in two steps. This is all is being done when the tester is testing the application using White box testing techniques.

- **Why is it essential to setup criteria for testing? List any three criteria in different situations**.

(Need to setup criteria for testing - 2 Marks, 3 criteria's - 2 Marks)

Ans: There is a need to setup criteria for testing because:

- An early start to testing reduces the cost, time to rework and error free software that is delivered to the client.

- Software Development Life Cycle (SDLC) testing can be started from the Requirements Gathering phase and lasts till the deployment of the software.

- It also depends on the development model that is being used. For example in Water fall model formal testing is conducted in the Testing phase, but in incremental model, testing is performed at the end of every increment/iteration and at the end the whole application is tested.

- Testing is done in different forms at every phase of SDLC like during Requirement gathering phase, the analysis and verification of requirements are also considered testing.

- Reviewing the design in the design phase with intent to improve the design is also considered as testing.

- Testing performed by a developer on completion of the code is also categorized as Unit type of testing.

Any 3 criteria's in different situation are :

- Start with the static white box testing procedure when the specifications of the software to be developed are ready.

- Use the code coverage analyzer to test whether the whole code is getting executed and covered.

3. Perform unit testing as soon as one of the unit or sub module in the software is ready.

Q List & explain techniques of finding bugs/ defects.

Ans: List of techniques:

- Static testing
- Dynamic testing 3.Operational testing

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

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

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

OR

Techniques to find defects are:

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

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

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

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

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

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

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

Q **Explain V-model with labelled diagram.**

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-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:**

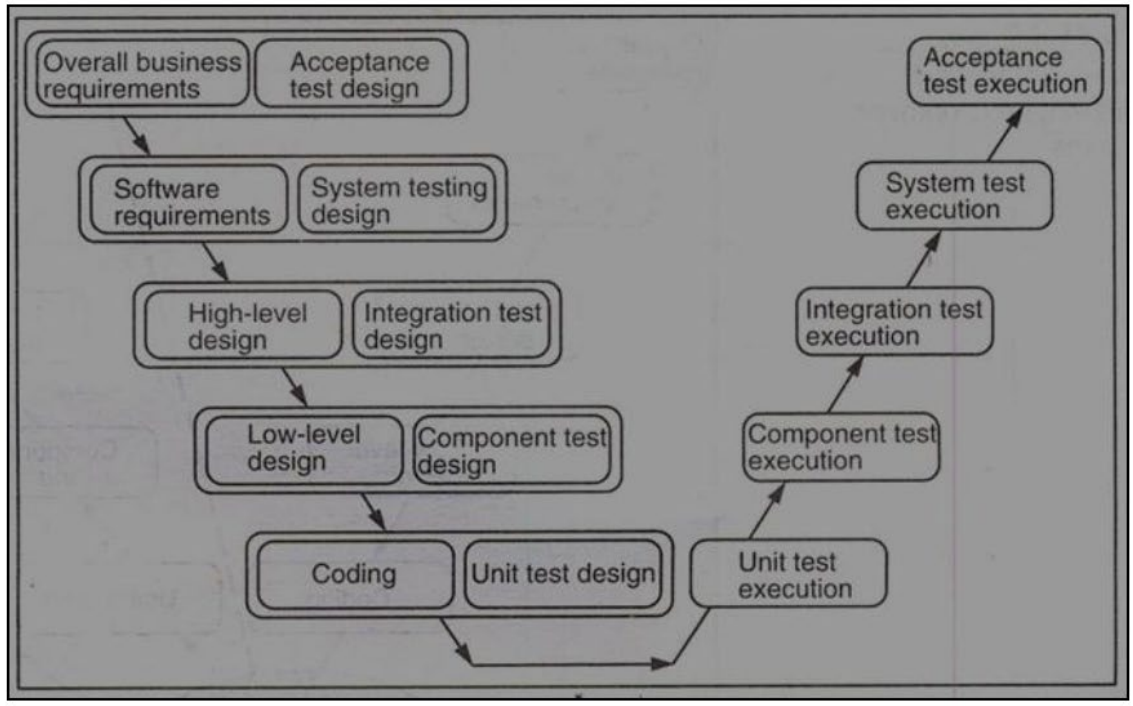
- **Overall Business Requirement:** In this first phase of the development cycle , the product requirements are understood from customer perspective. This phase involves detailed communication with the customer to understand his expectations and exact requirements. The acceptance test design planning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

- **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 requirements. Coding is done based on the coding guidelines & standards. **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

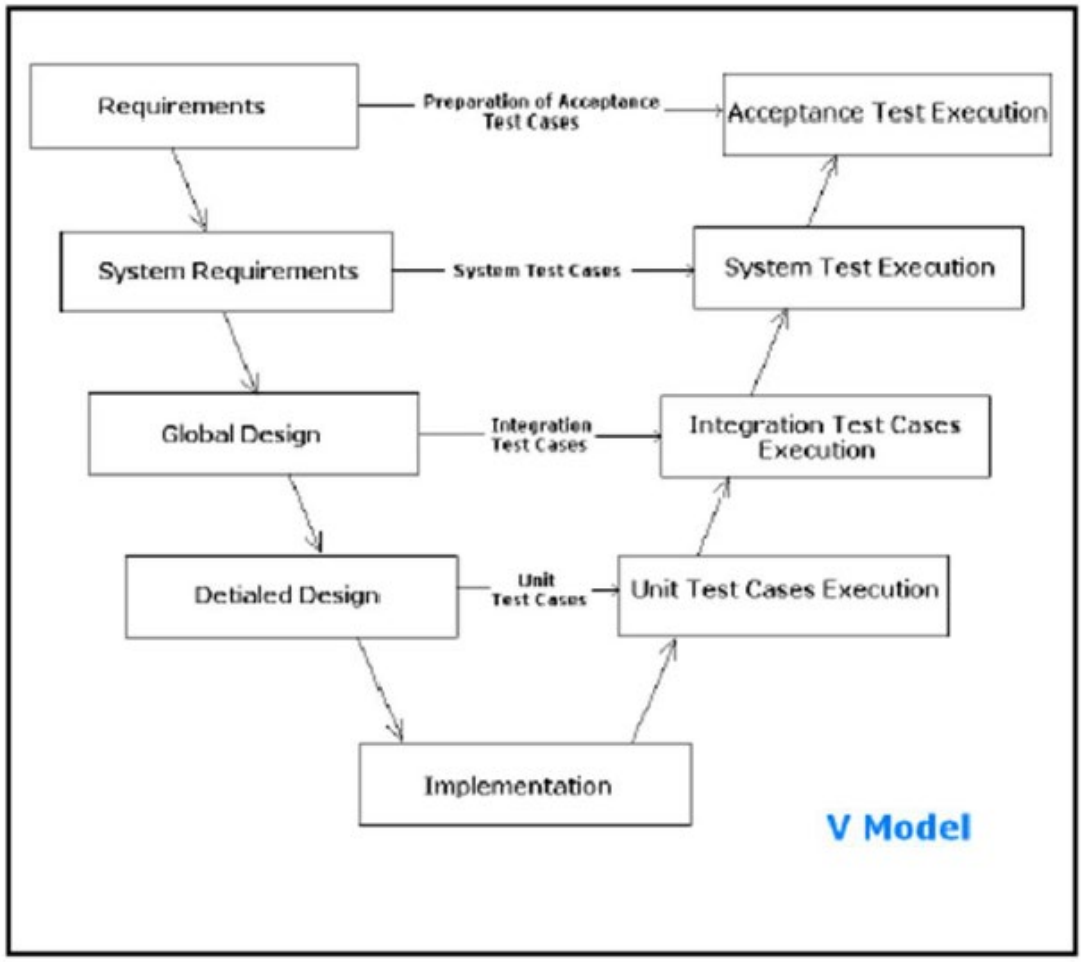
4. System Testing: It is associated with system design phase. It checks the entire

- system functionality & the communication of the system under development with external systems. Most of the software & hardware compatibility issues can be uncovered using system test execution.

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



OR



Q **Explain the terms Mistake, Error, Defect, Bug, Fault and Failure in relation with software testing.**

Ans: The various terms related to software failure with respect to the area of application are listed as **Defect, Variance, Fault, Failure, Problem, Inconsistency, Error, Feature, Incident, Bug, and Anomaly.**

Failure: the inability of a system or component to perform its required functions within specified performance requirements.

- **Fault:** An incorrect step, process, or data definition in a computer program.
- **Error:** A human action that produces an incorrect result.
- An **error** can be a **grammatical error** in one or more of the code lines, or a **logical error** in carrying out one or more of the client's requirements.
- Not all software errors become **software faults.** in some cases, the software error can cause improper functioning of the software. In many other cases, erroneous code lines will not affect the functionality of the software as a whole.
- **A failure** is said to occur whenever the external behaviour of a system does not conform to that prescribed in the system specification. A software fault becomes a software failure only when it is ―activated‖

The various terms related to software failure with respect to the area of application are listed as **Defect, Variance, Fault, Failure, Problem, Inconsistency, Error, Feature, Incident, Bug, and Anomaly.**

Problem, error, and bug are probably the most generic terms used.

1. • Anomaly, incident, and variance don't sound quite so negative and infer more unintended

2. operation than an all-out failure.
3. • Fault, failure, and defect tend to imply a condition that‘s really severe, maybe even dangerous. It doesn‘t sound right to call an incorrectly colored icon a fault. These words also tend to imply blame: ―It‘s his fault that the software failed.
4. • As all the words sound the same they are distinguished based on the severity and the area in which the software failure has occurred.
5. • When we run a program the error that we get during execution is termed on the basis of runtime error, compile time error, computational error, and assignment error.
6. • The error can be removed by debugging, if not resolved leads to a problem and if the problem becomes large leads to software failure.
7. • A bug can be defined as the initiation of error or a problem due to which fault, failure, incident or an anomaly occurs.
8. • The program to find the factorial of a number given below lists few errors problem and failure in a program.

For example

```
#include<stdio.h>
void main()
{
    int i , fact, n;
    printf(―enter the number ―);
    scanf(―%d‖,&n);
    for(i =1 ;i <=n;i++)
    fact = fact * i;
    printf (―the factorial of
    a number is ‖%d‖, fact);
}
```

As in line number 4 the fact is not initialized to 1, so it takes garbage value and gives a wrong output, this is an example of a bug. If fact is initialized to zero (fact = 0) than the output will be zero as anything multiplied by zero will give the output as zero. This is a bug which can be removed by initializing fact = 1 during initializing. As the fact is declared as integer, for the number till 7! will work perfectly. When the number entered is 8, the output is garbage value as the integer limit is from – 32767 to +32768, so in declaration change the initialization to long int fact.

Q **Describe inspection process performed under white box testing.**

Ans: Inspections are the most formal type of reviews in white box testing.
They are highly structured and require training for each participant.
Inspections are different from peer reviews and walkthroughs in that the person who presents the code, the presenter or reader, isn't the original programmer. These forces someone else to learn and understand the material being presented, potentially giving a different slant and interpretation at the inspection meeting. The other participants are called inspectors. Each is tasked with reviewing the code from a different perspective, such as a user, a tester, or a product support person. This helps bring different views of the product under review and very often identifies different bugs. One inspector is even tasked with reviewing the code backward—that is, from the end to the beginning—to make sure that the material is covered evenly and completely.

Q **Define software testing. List all skills of software tester.**

Ans: Software Testing is the process of executing a program with the intent of finding errors. A successful test is one that uncovers an as-yet-undiscovered error. Testing can show the presence of bugs but never their absence. Testing is a support function that helps developers look good by finding their mistakes before anyone else does. Execution of a work product

with intent to find a defect**.** Prevents defects**.**
Different Skills of software tester :

- Communication skills
- Domain knowledge
- Desire to learn
- Technical skills
- Analytical skills
- Planning
- Integrity
- Curiosity
- Think from users perspective
- Be a good judge for product

Q **Describe positive testing & negative testing. Also write test cases for them.**

Ans: Software testing is process of Verification and Validation to check whether software application under test is working as expected. To test the application we need to give some input and check if getting result as per mentioned in the requirements or not. This testing activity is carried out to find the defects in the code & improve the quality of software application. Testing of application can be carried out in two different ways, Positive testing and Negative testing.
Positive Testing: Positive Testing is testing process where the system validated against the valid input data. In this testing tester always check for only valid set of values and check if a application behaves as expected with its expected inputs. The main intention of this testing is to check whether software application not showing error when not supposed to & showing error when supposed to. Such testing is to be carried out keeping positive point of view & only execute the positive scenario. Positive Testing always tries to prove that a given product and project always meets the requirements and specifications. Under Positive testing is test the normal day to day life scenarios and check the expected behavior of application.
Negative Testing: Negative Testing is testing process where the system validated against the invalid input data. A negative test checks if a application behaves as expected with its negative inputs. The main intention of this testing is to check whether software application not showing error when supposed to & showing error when not supposed to. Such testing is to be carried out keeping negative point of view & only execute the test cases for only invalid set of input data. Negative testing is a testing process to identify the inputs where

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

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

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

Some more scenarios for testing:

Positive Test Scenarios:

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

Negative Test scenarios:

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

Q **Describe quality assurance and quality control.**

Ans: Quality Assurance:

i. It is Process oriented activities.

ii. A part of quality management focused on providing confidence that quality requirements will be fulfilled.

iii. All the planned and systematic activities implemented within the quality system that can be demonstrated to provide confidence that a product or service will fulfill requirements for quality

iv. Quality Assurance is fundamentally focused on planning and documenting those processes to assure quality including things such as quality plans and inspection and test plans.

v. Quality Assurance is a system for evaluating performance, service, of the quality of a

product against a system, standard or specified requirement for customers.
vi. Quality Assurance is a complete system to assure the quality of products or services. It is not only a process, but a complete system including also control. It is a way of management.

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

Q **What is Black Box testing? List any four techniques of Black Box testing.**

Ans: Black Box testing involves looking at the specifications and does not require examining the code of the program. It is done from customer‟s viewpoint. The testers know the input and expected output. They will check whether with given input they are getting expected output or not.

Different techniques of Black Box test are:

1. Requirement base testing
2. Positive negative testing
3. Boundary value analysis
4. Decision tables
5. Equivalence partitioning
6. State based testing
7. Compatibility testing
8. User documentation testing
9. Domain testing

Q With the help of example explain Boundary Value Analysis.

Ans: 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 the boundary. If the input data is used within the boundary value

limits, then it is said to be Positive Testing. If the input data is picked outside the boundary value limits, then it is said to be Negative Testing.

- Boundary value analysis is another black box test design technique and it is used to find the errors at boundaries of input domain rather than finding those errors in the centerof 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 othernumbers 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.

Q **State the different errors being found by the static and dynamic black box testing.**

Ans: **Errors found by static black box testing :**

Testing the specification is static black-box testing. The specification is a document, not an executing program, so it's considered static. It's also something that was created using data from many sources usability studies, focus groups, marketing input, and so on. Teh tester doesn't necessarily need to know how or why that information was obtained or the details of the process used to obtain it, just that it's been boiled down into a product specification. Tester can then take that document, perform static black-box testing, and carefully examine it for bugs.

The development team may emphasize diagrams over words or it may use a self-documenting computer language. It performs this with the help of :

- Performing a High-Level Review of the Specification
- Low-Level Specification Test Techniques

Overall all the errors in the specifications are found by this testing technic. Word to word the specifications are scanned and are understood to get the clear meaining of them.

Errors found by dynamic black box testing:

Dynamic Black-Box Testing is as good as Testing the Software While Blindfolded.

Testing software without having an insight into the details of underlying code is dynamic black- box testing. It's dynamic because the program is running you're using it as a customer would. And, it's black-box because you're testing it without knowing exactly how it works with blinders on.

The testers keep on entering inputs, receiving outputs, and checking the results. Another name commonly used for dynamic black-box testing is behavioral testing because its testing how the software actually behaves when it's used.

1 To do this effectively requires some definition of what the software does namely, a requirements document or product specification. Tester doesn't need to be told what happens inside the software "box" , tester just need to know that inputting A outputs B or that performing operation C results in D. A good product spec will provide tester with these details.

2 Once tester know the ins and outs of the software tester is about to test, testers next step is to start defining the test cases. Test cases are the specific inputs that he will try and the procedures that he will follow when he tests the software.

3 Partitioning of the data effectively is done for the software and checked.

4 Data testing is done with respect to boundry conditions , boundry value analysis, sub boundry conditions.

5 State testing is done by giving the inputs and gaining the outputs.

6 It tests the software‘s logic flow. Generates the state transtition maps.

7 Software is tested for load and stress conditions.

Q **Describe structural walk through under static testing.**

Ans: One of the static testing methods is structural walkthrough. In walkthroughs, a set of people look at the program code and raise questions for the author. The author explains the logic of the code and answers the questions. If the author is unable to answer some questions, he or she then takes those questions and finds their answers.

(i) Walkthroughs are the next step up in formality from peer reviews.

(ii) In a walkthrough, the programmer who wrote the code formally presents (walks through) it to a small group of five or so other programmers and testers.

(iii)The reviewers should receive copies of the software in advance of the review so they can examine it and write comments and questions that they want to ask at the review.

(iv) Having at least one senior programmer as a reviewer is very important.

Q) **Which are the various hardware and software required / recommended by project manager?**

Ans: 1. At the most basic level, project management products will help your organization to manage projects from start to finish, and allow employees at different levels to have an input into the process.

2. Project management software has been around for a number of years now and as a result, it does far more than just manage the projects themselves.

3. Project applications can also carry out scheduling, cost control and budget management, resource allocation, collaboration, communication, quality management and documentation or administration.

4. The aim with these is to handle all aspects and complexities of larger projects and help keep costs down.

Types and Levels of Testing

Q1) **State and explain top-down approach of integration testing with diagram.**

Ans: The strategy in top-down integration is look at the design hierarchy from top to bottom. Start with the high - level modules and move downward through the design hierarchy.

Modules subordinate to the top modules are integrated in the following two ways:

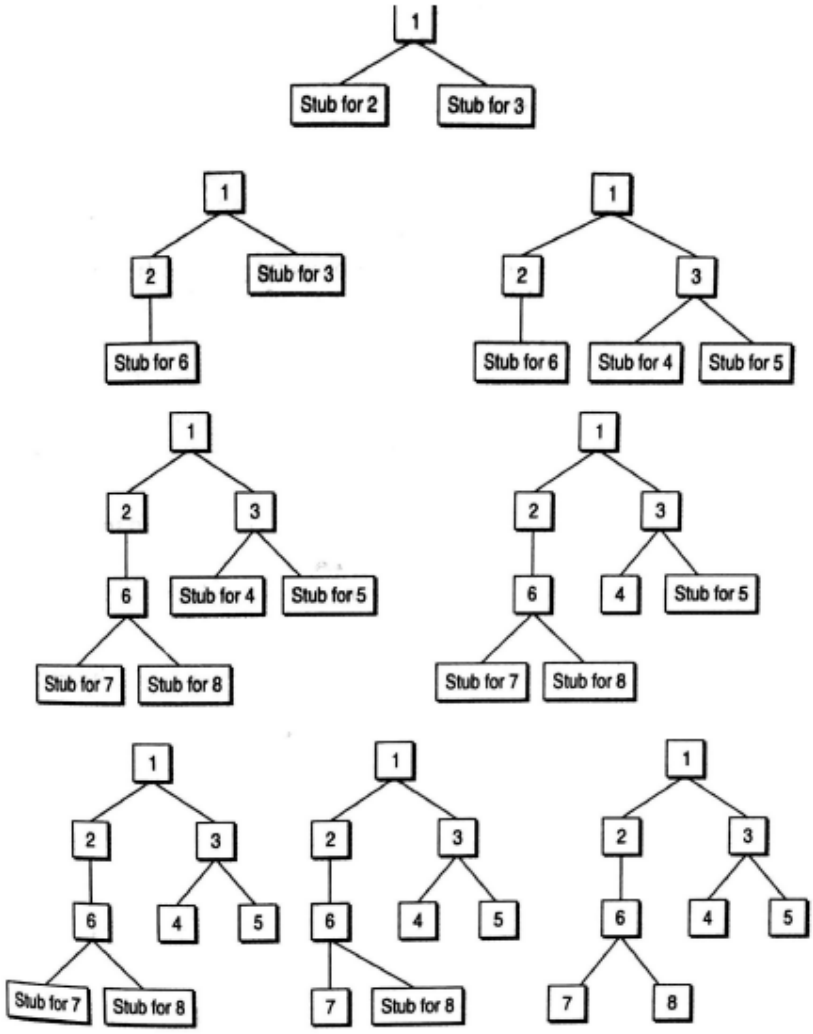
1. Depth first Integration: In this type, all modules on major control path of the design hierarchy are integrated first. In this example shown in fig. modules 1, 2, 6, 7/8 will be integrated first. Next, modules 1, 3, 4/5 will be integrated.

2. Breadth first Integration: In this type, all modules directly subordinate at each level, moving across the design hierarchy horizontally, are integrated first. In the example shown in fig.modules 2 and 3 will be integrated first. Next, modules 6,4 and 5 will be integrated . Modules 7 and 8 will be integrated last.

Procedure:

The procedure for Top-Down integration process is discussed in the following steps:

1. Start with the top or initial module in the software. Substitute the stubs for all the subordinate of the top module. Test the top module.
2. After testing the top module, stubs are replaced one at a time with the actual modules for integration.
3. Perform testing on this recent integrated environment.
4. Regression testing may be conducted to ensure that new errors have not appeared.
5. Repeat steps 2-4 for whole design hierarchy.



Q **Which are different methods of object oriented testing? Explain any one in detail.**

Ans: Traditional testing techniques can be adopted in Object Oriented environment by using the following techniques:

3. Method testing
4. Class testing
5. Interaction testing
6. System testing
7. Acceptance testing

Integration Testing: Object Orientation does not have a hierarchical control structure so conventional top-down and bottom up integration tests have little meaning. Integration testing can be applied in three different incremental strategies:

8. Thread-based testing, which integrates classes required to respond to one input or event.

9. Use-based testing, which integrates classes required by one use case. • Cluster testing, which integrates classes required to demonstrate one collaboration. Integration testing is performed using the following methods:

10. For each client class, use the list of class methods to generate a series of random test sequences. Methods will send messages to other server classes.

11. For each message that is generated, determine the collaborating class and the corresponding method in the server object.

12. For each method in the server object (that has been invoked by messages sent from the client object), determine the messages that it transmits.

13. For each of the messages, determine the next level of methods that are invoked and add these into the test sequence.

Q **Illustrate process of bi-directional integration testing. State its two advantages & disadvantages.**

1. **Ans:** 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.

OR

Process of Bidirectional testing:

6. Bottom up testing starts from middle layer and goes upward to the top layer. For a very big system, bottom up approach starts at a subsystem level and goes upwards.
7. Top down testing starts from the middle layer and goes downward. For a very big system, top down approach, starts at subsystem level and goes downwards.
8. Big band approach is followed for middle layer. From this layer, bottom up approach goes upwards and top down approach goes downwards.

Advantages:

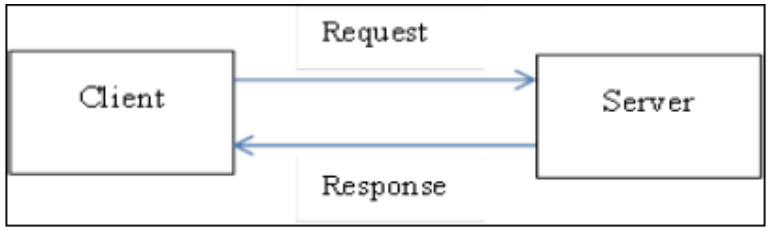
1. This approach is useful is useful for very large projects having several projects. When development follows a spiral model and module itself is as large as a system.
2. Both top down and bottom up approach starts at the start of the schedule.
3. It needs more resources and big teams for performing both, methods of testing at a time or one after the other.

Disadvantages:

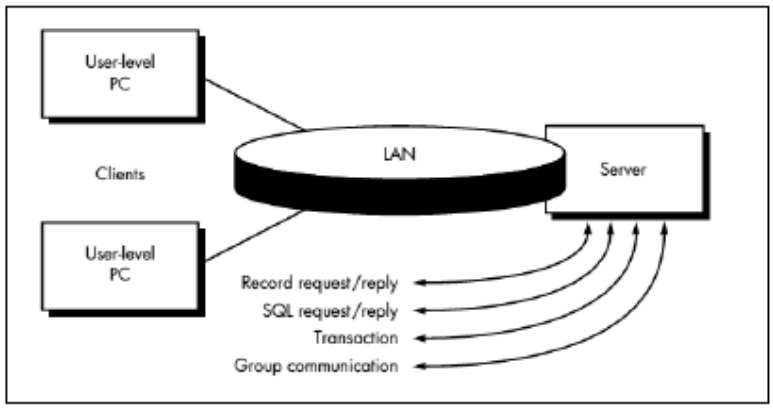
1. It represents very high cost of testing as lot of testing is done.
2. It cannot be used for smaller systems with huge interdependence between different modules.
3. Different skill tests are required for testers at different level as modules are separate systems handling separate domains.

Q **With the help of diagram describe client-server testing.**

Ans:



OR



In Client-server testing there are several clients communicating with the server.

14. Multiple users can access the system at a time and they can communicate with the server.
15. Configuration of client is known to the server with certainty.
16. Client and server are connected by real connection. 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, where as testing client a server simulator, and to test network both simulators are used at a time.

17. **Integration testing:** After successful testing of server, client and network, they are brought together to form system testing.
18. **Performance testing:** System performance is tested when number 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.
19. **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.
20. **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.
21. **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 over an extended period to understand if service level of network and server deteriorates over time due to some reasons like memory leakage.
22. **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.

Q **List any four features of client server application. Explain any four testing approaches of client – server testing.**

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

The approaches used for client server testing are

9. **User interface testing**: User interface testing, a testing technique used to identify the presence of defects is a product/software under test by using Graphical user interface [GUI].GUI Testing - Characteristics:
   - GUI is a hierarchical, graphical front end to the application, contains graphical objects with a set of properties.
   - During execution, the values of the properties of each objects of a GUI define the GUI state.
   - It has capabilities to exercise GUI events like key press/mouse click. iv)Able to provide inputs to the GUI Objects.

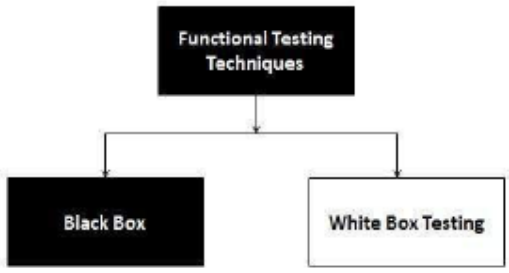
v) To check the GUI representations to see if they are consistent with the expected ones. vi)It strongly depends on the used technology.

2. **Manual testing**: Manual testing is a testing process that is carried out manually to find defects without the usage of tools or automation scripting. A test plan document is prepared that acts as a guide to the testing process to have the complete test coverage. Following are the testing techniques that are performed manually during the test life cycle are Acceptance Testing, White Box Testing, Black Box Testing, Unit Testing,

System Testing, Integration Testing.

3. Functional testing: Functional Testing is a testing technique that is used to test the features/functionality of the system or Software, should cover all the scenarios including failure paths and boundary cases.

There are two major Functional Testing techniques as shown below:



4.Compatibility testing: Compatibility testing is a non-functional testing conducted on the application to evaluate the application's compatibility within different environments. It can be of two types - forward compatibility testing and backward compatibility testing.

Operating system Compatibility Testing - Linux , Mac OS, Windows Database Compatibility Testing - Oracle SQL Server
Browser Compatibility Testing - IE , Chrome, Firefox
Other System Software - Web server, networking/ messaging tool, etc.

Q **Explain the Integration Testing and its two types. In each type, explain with example the steps of integration.**

Ans: Testing that occurs at the lowest level is called unit testing or module testing. As the units are tested and the low-level bugs are found and fixed, they are integrated and integration testing is performed against groups of modules. This process of incremental testing continues, putting together more and more pieces of the software until the entire product or at least a major portion of it is tested at once in a process called system testing.

With this testing strategy, it's much easier to isolate bugs. When a problem is found at the unit level, the problem must be in that unit. If a bug is found when multiple units are integrated, it must be related to how the modules interact. Of course, there are exceptions to this, but by and large, testing and debugging is much more efficient than testing everything at once.

Types of Integration testing:

1). Top down Testing: In this approach testing is conducted from main module to sub module. If the sub module is not developed a temporary program called STUB is used for simulate the sub module.

Advantages

Advantageous if major flaws occur toward the top of the program. Once the I/O functions are added, representation of test cases is easier. Early skeletal Program allows demonstrations and boosts morale.

Disadvantages:

Stub modules must be produced

Stub Modules are often more complicated than they first appear to be.

Before the I/O functions are added, representation of test cases in stubs can be difficult. Test conditions may be impossible, or very difficult, to create.

Observation of test output is more difficult.

Allows one to think that design and testing can be overlapped. Induces one to defer completion of the testing of certain modules.

27. **Bottom up testing:** In this approach testing is conducted from sub module to main module, if the main module is not developed a temporary program called DRIVERS is used to simulate the main module.

Advantages:

- Advantageous if major flaws occur toward the bottom of the program.
- Test conditions are easier to create.
- Observation of test results is easier.

Disadvantages:

- Driver Modules must be produced.
- The program as an entity does not exist until the last module is added.

28. **Bi-Directional Integration.**

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

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.

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.

Using stubs, it tests the user interface in isolation as well as tests the very lowest level functions using drivers. Bi-directional Integration testing combines bottom-up and top-down testing

Bottom-up testing is a process where lower level modules are integrated and then tested.

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.

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.

Top down testing helps developers find the missing branch link easily.

4). Incremental Integration.

After unit testing is completed, developer performs integration testing.

It is the process of verifying the interfaces and interaction between modules.

While integrating, there are lots of techniques used by developers and one of them is the incremental approach.

In Incremental integration testing, the developers integrate the modules one by one using stubs or drivers to uncover the defects.

This approach is known as incremental integration testing.

To the contrary, big bang is one other integration testing technique, where all the modules are integrated in one shot.

Features

1. Each Module provides a definitive role to play in the project/product structure
2. Each Module has clearly defined dependencies some of which can be known only at the runtime.
3. The incremental integration testing's greater advantage is that the defects are found early in a smaller assembly when it is relatively easy to detect the root cause of the same.
4. A disadvantage is that it can be time-consuming since stubs and drivers have to be developed for performing these tests.

5). Non- Incremental Integration.

The non-incremental approach is also known as –Big-Bang‖ Testing.

Big Bang Integration Testing is an integration testing strategy wherein all units are linked at once, resulting in a complete system.

When this type of testing strategy is adopted, it is difficult to isolate any errors found, because attention is not paid to verifying the interfaces across individual units.

Q Describe any four testing approaches of web application.

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

Web Application Testing - Techniques:

1. **Functionality Testing** - The below are some of the checks that are performed but not limited to the below list:

   Verify there is no dead page or invalid redirects. First check all the validations on each field.

   Wrong inputs to perform negative testing. Verify the workflow of the system.

   Verify the data integrity.

2. **Usability testing -** To verify how the application is easy to use with. Test the navigation and controls.

   Content checking. Check for user intuition.

3. **Interface testing -** Performed to verify the interface and the dataflow from one system to other.
4. **Compatibility testing-** Compatibility testing is performed based on the context of the application.

   Browser compatibility Operating system compatibility

   Compatible to various devices like notebook, mobile, etc.

5. **Performance testing -** Performed to verify the server response time and throughput under various load conditions.

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.

Soak testing - Soak Testing also known as endurance testing, is performed to determine the system parameters under continuous expected load. During soak tests the parameters such as memory utilization

is monitored to detect memory leaks or other performance issues. The main aim is to discover the system's performance under sustained use.

Spike testing -Spike testing is performed by increasing the number of users suddenly by a very large amount and measuring the performance of the system. The main aim is to determine whether the system will be able to sustain the work load.

6. **Security testing -** Performed to verify if the application is secured on web as data theft and unauthorized access are more common issues and below are some of the techniques to verify the security level of the system.

Injection
Broken Authentication and Session Management Cross-Site
Scripting (XSS)
Insecure Direct Object References Security
Misconfiguration
Sensitive Data Exposure
Missing Function Level Access Control
Cross-Site Request Forgery (CSRF)
Using Components with Known Vulnerabilities
Invalidated Redirects and Forwards

Q **Write important six test cases for the 'Login Form' of the Facebook website.**

Ans:

Test cases for facebook login are as given below:

Step	Test step	Test data	Expected output	Actual output	Status
1	Navigate to login page of facebook	---	---	---	Pass
2	Provide valid username	Username abc@yahoo.co.in	Shall accept the username	Accepted user name	Pass
3	Provide password	Password co6g1234	Shall accept the password	Accepted the password	Pass
4	Click on submit	Press submit button	User should be able to login successfully	User successfully logged in	Pass
5	Go to the home page	Click on home button	Should display home page	Home page of the user displayed	Pass
6	Write the status	Type in the status in the area provided and press post	Should post the message typed in and the status of user should change	Status changed successfully	Pass

Q **Prepare six test cases for home page of marketing site www.flipkart.com.**

Ans:

Sr. No.	**Test Case -ID**	**Test case Objec tive**	**Prereq uisite**	**Steps**	**Input data**	**Expected Result**	**Actual Result**	**Remarks/ Status**

1	**TC-1**	To check interes t	Whether internet is available or not?	Test internet connection	www.flipkart.com	Site home page display should display on screen	Hom e page displ ayed	Tes t to fail
2	**TC-2**	User name	correct and valid user name should be registere d on flipkart	Ty pe cor rect and vali d use r na me	User nam e give n by flipk art	Should enter valid the user name	Goes to next page if user id and pass word verifi ed and valid	Tes t to pas s
							ated	
3	**TC-3**	Passwo rd	correct and valid passwor d should be validate and verified on flipkart site	Ty pe cor rect and vali d pas sw ord	Pass wor d selec ted by user and valid ated by flipk art	Should enter valid passw ord	Goes to next page if user id and pass word verifi ed and valid ated	Tes t to pas s
4	**TC-4**	To check wheth er site home page opens or not	To displa y flipkar t home page websit e on screen	Type site proper addres s as www.f lipkart .com	www.fli pkart .com	Site home page display should display on screen	Hom e page displ ayed	Tes t to fail
5	**TC-5**	To search the produc t	Search the product	Click on produc t link	Mou se rollo ver and click	Shall display availab lity of product	Sho w avail ablit y of prod uct	Tes t to pas s

6	**TC-6**	Product details	Product shall be available on site	Click on product link	Mouse rollover and click	Shall display product details	displays product details of other product	Test to fail
7	**TC-7**	Product details	Product shall be available on site	Click on product link	Mouse rollover and click	Shall display product details	displays product details of product	Test to pass
8	**TC-8**	Search for offers	Offer should be valid for that day	Click on Offer link	Select Offer icon and click	Shall display offer details	displays offer details of selected product	Test to pass

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

Q **Describe any two special tests in testing process.**

Ans: Smoke Testing: is a testing technique that is inspired from hardware testing, which checks for the smoke from the hardware components once the hardware's power is switched on. ii. In Software testing context, smoke testing refers to testing the basic functionality of the build. iii. If the Test fails, build is declared as unstable and it is NOT tested anymore until the smoke test of the build passes.

Smoke Testing - Features:

- Identifying the business critical functionalities that a product must satisfy.
  ii. Designing and executing the basic functionalities of the application.
  iii. Ensuring that the smoke test passes each and every build in order to proceed with the testing.
  iv. Smoke Tests enables uncovering obvious errors which saves time and effort of test team.
  v. Smoke Tests can be manual or automated.

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

30. **Regression Testing:** Regression testing a black box testing technique that consists of re- executing those tests that are impacted by the code changes. ii. These tests should be executed as often as

possible throughout the software development life cycle. Types of Regression Tests: i. Final Regression Tests: - A "final regression testing" is performed to validate the build that hasn't changed for a period of time. This build is deployed or shipped to customers. ii. Regression Tests: - A normal regression testing is performed to verify if the build has NOT broken any other parts of the application by the recent code changes for defect fixing or for enhancement.

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

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

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

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

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

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

Web Application Testing Techniques:

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

Q **Describe use of load testing and security testing in performance testing of facility of online result display of MSBTE.**

Ans: : In case of testing of facility of online result display of MSBTE, Performance testing, will be a non-functional testing technique performed to determine the system parameters in terms of responsiveness and stability under various workload on the website. It will measures the quality attributes of the

Load testing - It is the simplest form of testing conducted to understand the behavior of the system under a specific load. Load testing will result in measuring number of critical transactions and load on the database, application server, etc., are also monitored. It will show how the software will respond if the number of students checking the result at the same time are more than specified.

Response time of the system, throughput, resource utilization, and Maximum user load will be checked by this type of testing.

Security Testing: Security testing is a testing technique to determine if an information system protects data and maintains functionality as intended. It also aims at verifying 6 basic principles as Confidentiality of the MSBTE online result display system, Integrity of the software, authentication of the user who logs in to the system to check the result with the help of valid seat number and enrollment number, authorization, availability , and Non-repudiation of the system.

Q Describe Graphical User Interface (GUI) testing and its important traits.

Ans:

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

GUI Testing Guidelines

- Check Screen Validations
- Verify All Navigations
- Check usability Conditions
- Verify Data Integrity
- Verify the object states
- Verify the date Field and Numeric Field Formats

GUI Automation Tools

Following are some of the open source GUI automation tools in the market:

Product	**Licensed Under**	**URL**
AutoHotkey	GPL	http://www.autohotkey.com/
Selenium	Apache	http://docs.seleniumhq.org/
Sikuli	MIT	http://sikuli.org
Robot Framework	Apache	www.robotframework.org
Water	BSD	http://www.watir.com/
Dojo Toolkit	BSD	http://dojotoolkit.org/

GUI Automation

Q **How performance testing is performed? List steps involved in it?**

1. **Ans:** Performance testing, a non-functional testing technique performed to determine the system parameters in terms of responsiveness and stability under various workload.

2. Performance testing measures the quality attributes of the system, such as scalability, reliability and resource usage.

Techniques are:

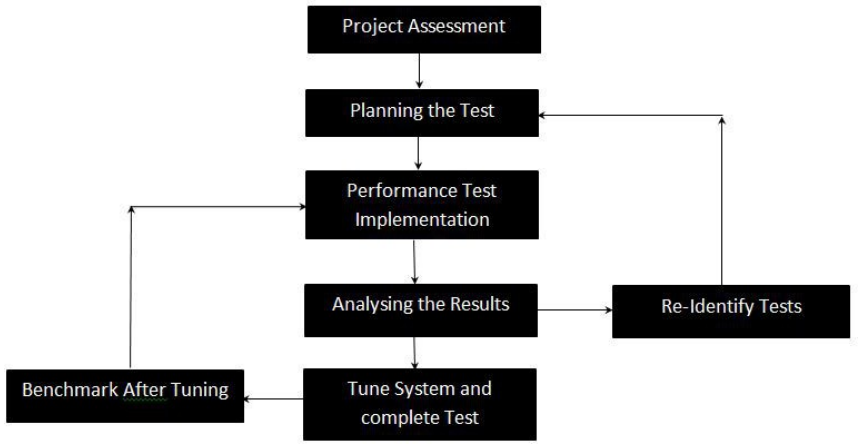
1. **Load testing -** It is the simplest form of testing conducted to understand the behaviour of the system under a specific load. Load testing will result in measuring important business critical transactions and load on the database, application server, etc., are also monitored.

2. **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.

3. **Soak testing -** Soak Testing also known as endurance testing, is performed to determine the system parameters under continuous expected load. During soak tests the parameters such as memory utilization is monitored to detect memory leaks or other performance issues. The main aim is to discover the system's performance under sustained use.

4. **Spike testing -** Spike testing is performed by increasing the number of users suddenly by a very large amount and measuring the performance of the system. The main aim is to determine whether the system will be able to sustain the workload.

Performance Testing Process:



Q **Explain the performance testing and its criterias.**

Ans: **Performance testing**: Performance testing is intended to find whether the system meets its performance requirements under normal load or abnormal level of activities. Normal load must be defined by the requirement statement defined by the customer and system design implements them. Performance criteria must be expressed in numerical terms. Design verification can help in determining whether required measures have been taken to meet performance requirements or not. This is one area where verification does not work to that many extents and one needs to test it by actually performing the operation on the system. It can serve different purposes like it can demonstrate that the system meets performance criteria.

Criteria of Performance testing:

Stress Testing:

In stress testing the resources are used less than the requirement. If the system has limited resources available, the response of the system may deteriorate due to non-availability of the resources. It tries to break the system 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 behavior of the software under abnormal situations. In stress testing test cases are designed to execute the system in such a way that abnormal conditions.

Load Testing:

When a system is tested with a load that causes it to allocate its resources in maximum amounts. The idea is to create an environment more demanding than the application would experience under normal workloads. Load is varied from minimum to the maximum level the system can sustain without running out of resources. Load is being increased transactions may suffer excessive delays. Load testing involves simulating real-life user load for the target application. It helps to determine how application behaves when multiple users hits it simultaneously. Load testing can be done under controlled lab conditions to compare the capabilities of different
systems or to accurately measure the capabilities of a single system.

Q **What is load testing and stress testing? Describe with respect to system testing.**

Ans: **Stress testing** is testing the software under less than ideal conditions. So subject your software to low memory, low disk space, slow cpus, and slow modems and so on. Look at your software and determine what external resources and dependencies it has. Stress testing is simply limiting them to bare minimum. With stress testing you starve the software.

For e.g. Word processor software running on your computer with all available memory and disk space, it works fine. But if the system runs low on resources you had a greater potential to expect a bug. Setting the values to zero or near zero will make the software execute different path as it attempt to handle the tight constraint. Ideally the software would run without crashing or losing data.

Load testing is testing the software under customer expected load. In order to perform load testing on the software you feed it all that it can handle. Operate the software with largest possible data files. If the software operates on peripherals such as printer, or communication ports, connect as many as you can. If you are testing an internet server that can handle thousands of simultaneous connections, do it. With most software it is important for it to run over long periods. Some software's should be able to run forever without being restarted. So Time acts as a important variable.

Stress testing and load testing can be best applied with the help of automation tools**.** Stress testing and load testing are the types of performance testing.

The Microsoft stress utility program allows you to individually set the amounts of memory, disk space, files and other resources available to the software running on the machine.

Example: Open many number of browsers in the windows simultaneously.

Connect more than the specifies clients to the server.

Connect more than one printer to the system.

Q **Write a short note on :**

(iii)Recovery testing

(iv) Usability testing

Ans: iii**) Recovery testing:** Recovery testing is a type of non-functional testing. Recovery testing is done in order to check how fast and better the application can recover after it has gone through any type of crash or hardware failure etc.

- Recovery testing is the forced failure of the software in a variety of ways to verify that recovery is properly performed.
- Determining the feasibility of the recovery process.
- Verification of the backup facilities.
- Ensuring proper steps are documented to verify the compatibility of backup facilities.
- Providing Training within the team.
- Demonstrating the ability of the organization to recover from all critical failures.
- Maintaining and updating the recovery plan at regular intervals.

(iv)Usability testing: Usability testing, a non-functional testing technique that is a measure of how easily the system can be used by end users.It is difficult to evaluate and measure but can be evaluated based on the below parameters:

 Levels of Skill required learn/use the software. It should maintain the balance for both novice and expert user.

 Time required to get used to in using the software.

 The measure of increase in user productivity if any.

 Assessment of a user's attitude towards using the software.

 Usability testing, a non-functional testing technique that is a measure of how easily the system can be used by end users.

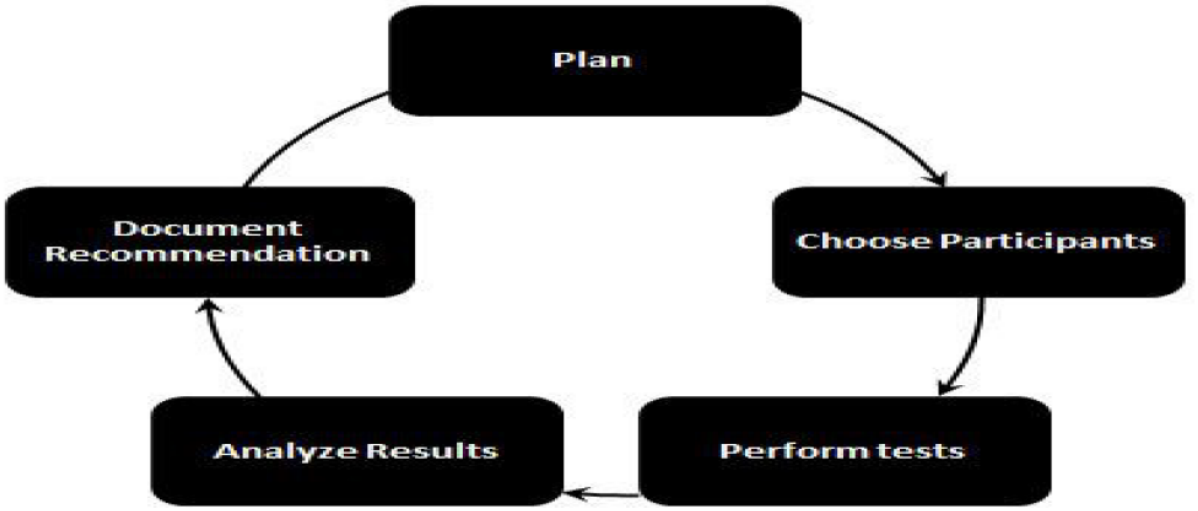
 It is difficult to evaluate and measure but can be evaluated based on the below parameters:

 Levels of Skill required learn/use the software. It should maintain the balance for both novice and expert user.

 Time required to get used to in using the software.

 The measure of increase in user productivity if any.

Usability Testing Process:



Q **List any four advantages of acceptance test before launching any software.**

1. **Ans:** 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 quality of 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

Q **Describe alpha testing with its entry & exit criteria.**

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

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

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

 Entry criteria, specifies when that phase can be started also included the inputs for the phase.

☐ Tasks or steps that need to be carried out in that phase, along with measurements that characterize the tasks.
☐ Verification**,** which specifies methods of checking that tasks have been carried out correctly.
☐ Clear entry criteria make sure that a given phase does not start prematurely.
☐ The verification for each phase helps to prevent defects. At least defects can be minimized.
Exit criteria, which stipulate the conditions under which one can consider the phases as done and included are the outputs for the phase.
a. Exit criteria may include:
1. All test plans have been run
2. All requirements coverage has been achieved.
3. All severe bugs are resolved.

OR

Entry Criteria for **Alpha testing:**

- Software requirements document or Business requirements specification
- Test Cases for all the requirements
- Testing Team with good knowledge about the software application
- Test Lab environment setup
- QA Build ready for execution
- Test Management tool for uploading test cases and logging defects
- Traceability Matrix to ensure that each design requirement has atleast one test case that verifies it

Exit Criteria for **Alpha testing:**

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

Q **Write 4 test cases for user login form.**

Ans:

TC_Id	TC_name	Steps	Input data	Expected result	Actual Result	Status
TC_01	User name	Enter the username in alphanumeric alphabets A-Z Number 0-9	“abc123”	It should accept the username	It is accepted username	Pass
TC_02	Password	Enter the password in alphanumeric alphabets A-Z Number 0-9	“abc123”	It should accept the password	It is accepted password	pass

TC_03	Submit	35. After valid username and password 36. Click on submit button		It should goes to next page	It is going to next page	pass
TC_04	Cancel	Click on cancel button		It should remain in login page with blank fields	It shows login page with blank fields	pass

Q **Explain the alpha testing. State its limitations.**

Ans: Alpha testing is done by the customers in development environment in front of the development team.

Limitations:

1. Data provided by the customer may not represent actual data or business data. If data is created by testers using any data definition technique, probability of such data occurring in real life must be checked and validated by the customer.
2. Laboratory environment may not represent real life environment.
3. Key users deployed by customer may not be the people who are going to use the system in reality and also may not be aware of actual working expectations from a new system.

Q **Compare alpha testing and beta testing.**

Ans:

Alpha Testing	Beta Testing
1. Performed at Developer_s site.	1. Performed at End user_s site.
2. Performed in controlled Environment as developer is present.	2. Performed in uncontrolled Environment as developer is not present.
3. Less probability of finding of errors as it is driven by developer.	3. High probability of finding errors as end user can use it the way he wants.
4. It is done during implementation phase of software	4. It is done as pre-release of software.
5. It is not considered as live application	5. It is considered as live application.
6. Less time consuming as developer can make necessary changes in given time.	6. More time consuming. As user has to report bugs if any via appropriate channel

Q **Explain the regression testing. State when the regression testing shall be done.**

Ans: 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.

Q **State any four objectives of user documentation testing. How these are useful in planning user documentation test? Mention any two benefits of user documentation testing.**

Ans: Documentation testing is a non-functional type of software testing.

1. It is a type of non-functional testing.
2. Any written or pictorial information describing, defining, specifying, reporting, or certifying activities, requirements, procedures, or results'. 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.
3. 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.
4. This is one of the most cost effective approaches to testing. If the documentation is not right: there will be major and costly problems. The documentation can be tested in a number of different ways to many different degrees of complexity. These range from running the documents through a spelling and grammar checking device, to manually reviewing the documentation to remove any ambiguity or inconsistency.
5. Documentation testing can start at the very beginning of the software process and hence save large amounts of money, since the earlier a defect is found the less it will cost to be fixed.

Benefits:

1. User documentation testing aids in highlighting problems over looked during reviews.
2. High quality user documentation ensures consistency of documentation & product, thus minimizing possible defects reported by customer. It also reduces the time taken for each support call.
3. Result in less difficult support call. When customer faithfully follows the instruction given in a document but is unable to achieve to desire result, it is frustrating and often this frustration shows up on the support staff. Ensuring that a product is tested to work as per the document and that it works correctly contributes to better customer satisfaction and better morale of support staff.
4. New programmers and testers who join a project group can use the documentation to learn the external functionality of the product.

Customers need less training and can proceed more quickly to advanced training & product usage if the user documentation is of high quality & is consistent with the product. Thus high-quality user documentation can result in a reduction of overall training costs for user organization.

Q **State purpose of code coverage. How it is used in analyzing coding of software?**

Ans: Purpose:

Code coverage is used to find out the percentage of code that is covered by testing.
It is used for analyzing the coding software: As product is realized in terms of program code we can run test cases to exercise the different parts of the code gets tested. Code coverage testing involves designing and executing test cases and finding out the percentage of the code that is covered by testing. It is found by adopting a technique called instrumentation of the code. This instrumented code can monitor and keep audit of what portions of code are covered. The tools also allow reporting on the portions of the code that are covered frequently, so that the critical or most-often portions of the code can be identified.

Code coverage testing is made up of the following types of coverage:

1. Statement coverage
2. Path coverage
3. Condition coverage

4. Function coverage
5. Branch coverage

Q **Explain concept of application of equivalence partitioning. How it is useful in result analysis of diploma?**

Ans: Equivalence partitioning is a software technique that involves identifying a small set of representative input values that produce as much different output condition as possible. This reduces the number of permutation & combination of input, output values used for testing, thereby increasing the coverage and reducing the effort involved in testing.

The set of input values that generate one single expected output is called a partition. When the behavior of the software is the same for a set of values, then the set is termed as equivalence class or partition.

Example

Marks	**result**
Under 40	fail
40-100	pass

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

Above	40 marks in subject	(valid input)
Between	40 and 100 marks	(valid input)
Below	40 marks	(invalid input)
	Negative marks	(Invalid Input)

Q **Describe Inspection under static testing.**

Ans: Under Static testing is to review the code without executing it. Inspection is the most formal method in static testing. This method can detect all faults, violations and other side effects.

OR

Inspection is formal review where people external to the testing team may be involved as inspectors. They are subject matter experts who review the work product.

In this:

1. Thorough preparation is required before an inspection/review
2. Enlisting multiple diverse views.
3. Assigning specific roles to the multiple participants
4. Going sequentially through the code in a structured manner.

There are four roles in inspection:

1. **Author** of the code: the person who had written the code
2. **Moderator:** who is expected to formally run the inspection according to the process?
3. **Inspectors:** are the people who actually provide review comments for the code.
4. **Scribe:** who takes detail notes during the inspection meeting and circulates them to the inspection team after the meeting.

The author or moderator selects review team. The inspection team assembles at the agreed time for inspection meeting. The moderator takes the team sequentially through the program code. If any defect is found they will classify it as minor or major. A scribe documents the

defects. For major defects the review team meets again to check whether the bugs are resolved or not.

Q **How to perform security testing? State elements of security testing.**

Ans: Security Testing: Testers must use a risk-based approach, By identifying risks and potential loss associated with those risks in the system and creating tests driven by those risks, the testers can properly focus on areas of code in which an attack is likely to succeed. Therefore risk analysis at the design level can help to identify potential security problems and their impacts. Once identified ranked, software risks can help guide software security. It is a type of non-functional testing. Security testing is basically a type of software testing that's done to check whether the application or the product is secured or not. It checks to see if the application is vulnerable to attacks, if anyone hack the system or login to the application without any authorization. It is a process to determine that an information system protects data and maintains functionality as intended. The security testing is performed to check whether there is any information leakage in the sense by encrypting the application or using wide range of software's and hardware's and firewall etc.

Software security is about making software behave in the presence of a malicious attack.

The six basic security concepts / elements that need to be covered by security testing are:

- confidentiality,
- integrity,
- authentication,
- availability,
- authorization and
- Non-repudiation.

Q Illustrate process of graph-based testing with suitable example.

Ans: i. Black-box methods based on the nature of the relationships (links) among the program objects (nodes), test cases are designed to traverse the entire graph

ii. Transaction flow testing – nodes represent steps in some transaction and links represent logical connections between steps that need to be validated

iii. Finite state modeling – nodes represent user observable states of the software and links represent transitions between states

iv. Data flow modeling – nodes are data objects and links are transformations from one data object to another

v. Timing modeling – nodes are program objects and links are sequential connections between these objects, link weights are required execution times.

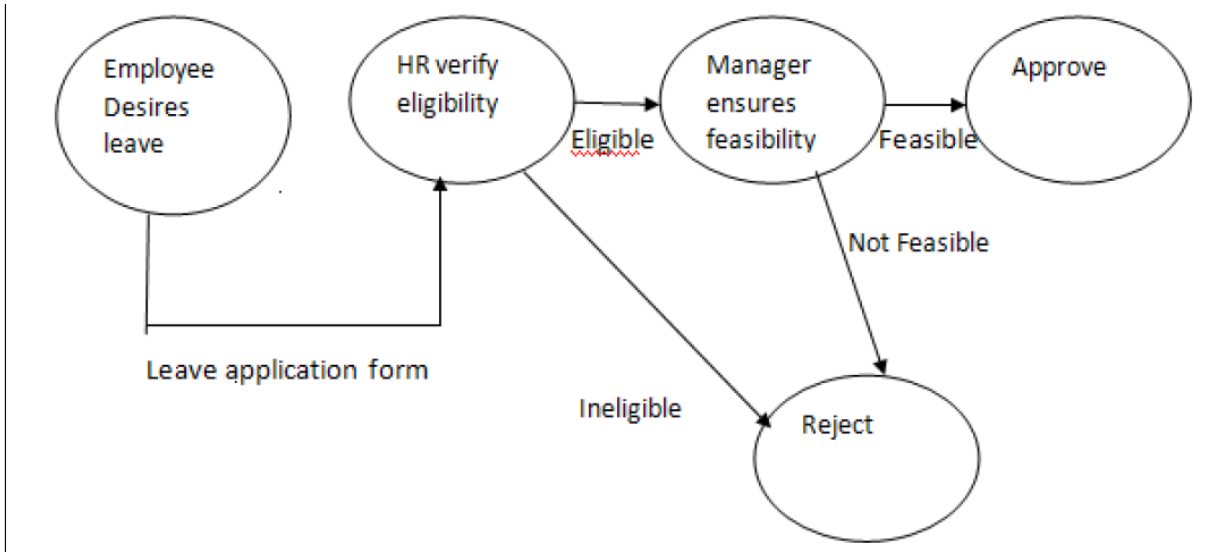
Steps in graph testing:

i. Build a graph model.

ii. Identify the test requirements.

iii. Select test paths to cover those requirements.

Derive test data so that those test paths can be executed.



Q **With the suitable example, explain how 'Basis Path Testing' is used to derive the code complexity for the testing.**

Ans: **Basic path testing is the structural testing technique:**

Path testing is based on control structure of the program for which flow graph is prepared

Path testing requires complete knowledge of the programs structure.

Path testing is closer to the developer and used by him to test his module.

The effectiveness of path testing gets reduced with te increase in size of software under test. Choose enough paths in a program such that maximum logic coverage is achieved.

Branch Coverage, code coverage, line coverage is also called as basis path testing. Attempting to cover all the paths in the software is called basis path testing. It's the actual structural testing which is the part of static white box testing. So many times static white box testing is called basis path testing. The simplest form of path testing is called branch coverage testing. To check all the possibilities of the boundary and the sub boundary conditions and it's branching on those values. Test coverage criteria requires enough test cases such that each condition in a decision takes on all possible outcomes at least once, and each point of entry to a program or subroutine is invoked at least once.

Every branch (decision) taken each way, true and false. It helps in validating all the branches in the code making sure that no branch leads to abnormal behavior of the application.

For example in the following code all the branches in the program are checked thoroughly. The decisions are evaluated and are tested whether are not these branches are taken.

```
#include<stdio.h>

void main()

{

int i , fact= 1, n;

printf("enter the number ");
scanf(-%d||, &n);

for(i =1 ;i <=n; i++)
fact = fact * i;

printf ("the factorial of a number is ",%d, fact);

}
```

Data Flow (Code Functional Testing)

Data flow coverage involves tracking a piece of data completely through the software. At the

unit test level this would just be through an individual module or function. The same tracking could be done through several integrated modules or even through the entire software product— although it would be more time consuming. During data flow, the check is made for the proper declaration of variables declared and the loops used are declared and used properly.

Line coverage or code coverage testing:

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

For example

```
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. }
```

Test Management

Q1) **Which features are included in test approach while planning test?**

Ans: Like any project, the testing also should be driven by a plan. The test plan acts as the anchor for the execution, tracking and reporting of the entire testing project. Activities of test plan:

1. Scope Management: Deciding what features to be tested and not to be tested.
2. Deciding Test approach /strategy: Which type of testing shall be done like configuration, integration, localization etc.
3. Setting up criteria for testing: There must be clear entry and exit criteria for different phases of testing. The test strategies for the various features and combinations determined how these features and combinations would be tested.
4. Identifying responsibilities, staffing and training needs
5. Identifying resource requirements
6. Identifying test deliverables
7. Testing tasks: size and effort estimation

Q **What is test planning and test management?**

Ans: **Test Planning**: Like any project, the testing also should be driven by a plan. The test plan acts as the anchor for the execution, tracking and reporting of the entire testing project. Activities of test plan:

8. Scope Management: Deciding what features to be tested and not to be tested.
9. Deciding Test approach /strategy: Which type of testing shall be done like configuration, integration, localization etc.
10. Setting up criteria for testing: There must be clear entry and exit criteria for different phases of testing. The test strategies for the various features and combinations determined how these features and combinations would be tested.
11. Identifying responsibilities, staffing and training needs

Test Management: It concerned with both test resource and test environment management. It is the role of test management to ensure that new or modified service products meet business requirements for which they have been developed or enhanced.

Q **Explain the risk management in the test planning.**

Ans: **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.

Q **State the contents of ‘Test Summary Report’ used in test reporting.**

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

1. Test incident report:

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

2. Test cycle report:

A test cycle entails planning and running certain test in cycle, each cycle using a different build of the product .As the product progresses through the various cycles it is expected to stabilize. Test cycle report gives
A summary of the activities carried out during that cycle.

Defects that are uncovered during that cycle based on severity and impact

Progress from the previous cycle to the current cycle in terms of defect fixed

Outstanding defects that not yet to be fixed in cycle

Any variation observed in effort or schedule

3. Test summary report:

The final step in a test cycle is to recommend the suitability of a product for release. A report that summarizes the result of a test cycle is the test summary report. **There are two types of test summary report:**
Phase wise test summary, which is produced at the end of every phase

Final test summary report.

A Summary report should present
Test Summary report Identifier

Description

Identify the test items being reported in this report with test id
1). Variances

Mention any deviation from test plans, test procedures, if any.
2). Summary of results

All the results are mentioned here with the resolved incidents and their solutions.
3). Comprehensive assessment and recommendation for release should include Fit for release assessment and recommendation of release

Q **What is test deliverables and milestones? Explain any four test deliverables.**
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

- 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.
- Test cases list the specific items that will be tested and describe the detailed steps that will be followed to verify the software.

- Bug reports describe the problems found as the test cases are followed. These could be done on paper but are often tracked in a database.
- 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.
- 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.

Q **State the contets of standard template of a test plan.**

Ans: **TEST PLAN TEMPLATE**

- Introduction

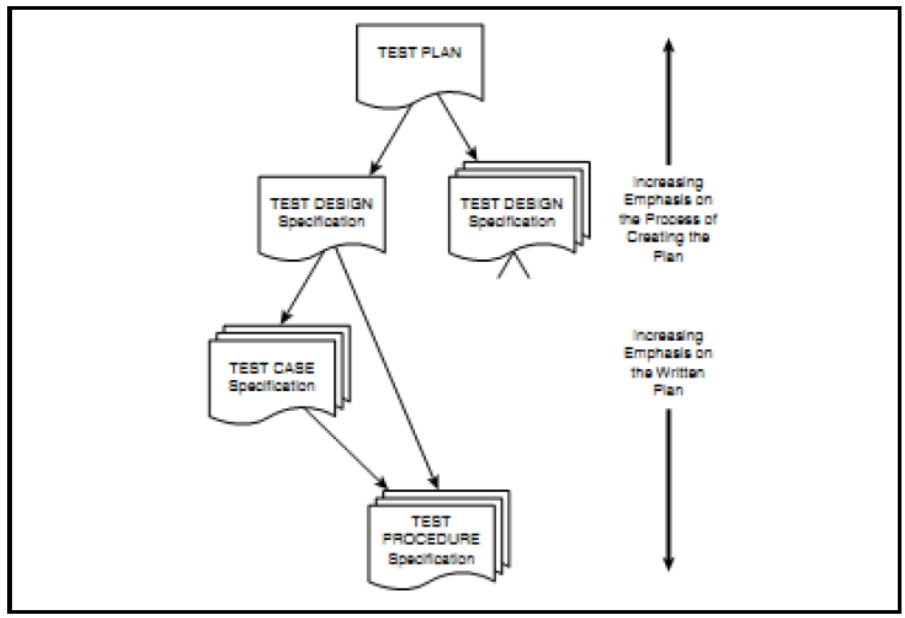
  1.1 Scope

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

- References
- Test Methodology and Strategy/Approach
- Test Criteria
  - Entry Criteria
  - Exit Criteria
  - Suspension Criteria
  - Resumption Criteria
- Assumptions, Dependencies, and Risks
  - Assumption
  - Dependencies
  - Risk and Risk Management Plans
- Estimations
  - Size Estimate
  - Effort Estimate
  - Schedule Estimate
- Test Deliverables and Milestones
- Responsibilities
- Resource Requirement
  - Hardware Resources
  - Software Resources
  - People Resources
  - Other Resources
- Training Requirements
  - Detail of Training Required
  - Possible Attendees
  - Any Constrains
- Defect Logging and Tracking Process
- Metrics Plan
- Product Release Criteria

Q **Explain the 'Test infrastructure' components with neat diagram.**
Ans:

Test infrasturcture componets diagram :



The top, or project level, test plan, the process of creating it is more important than the resulting document. The next three levels, the test design specification, the test case specification, and the test procedure specification are described in detail in the following sections.

As you can see in Figure, moving further away from the top-level test plan puts less emphasis on the process of creation and more on the resulting written document. The reason is that these plans become useful on a daily, sometimes hourly, basis by the testers performing

the testing. At the lowest level they become step-by-step instructions for executing a test, making it key that they're clear, concise, and organized how they got that way isn't nearly as important.

This standard is what many testing teams have adopted as their test planning documentation intentional or not—because it represents a logical and common-sense method for test planning. The important thing to realize about this standard is that unless tester is bound to follow it to the

letter because of the type of software he is testing or by your corporate or industry policy, tester should use it as a guideline and not a standard.

Test Design

The overall project test plan is written at a very high level. It breaks out the software into specific features and testable items and assigns them to individual testers, but it doesn't specify exactly how those features will be tested. There may be a general mention of using automation or black- box or white-box testing, but the test plan doesn't get into the details of exactly where and how they will be used. This next level of detail that defines the testing approach for individual software features is the test design specification.

Test Cases

Dissecting a specification, code, and software to derive the minimal amount of test cases that would effectively test the software. The test case specification –documents the actual values used for input along with the anticipated outputs. A test case also identifies any constraints on the test procedure resulting from use of that specific test case.‖ Essentially, the details of a test case should explain exactly what values or conditions will be sent to the software and what result is expected. It can be referenced by one or more test design specs and may reference more than one test procedure. The ANSI/IEEE 829 standard also lists some other important information that should be included:

- **Identifiers.**
- **Test item**.
- **Input specification.**

- **Output specification.**
- **Environmental needs.**
- **Special procedural requirements.**
- **Intercase dependencies**.
-

Test Procedures

After tester documents the test designs and test cases, what remains are the procedures that need to be followed to execute the test cases. The test procedure specification –identifies all the steps required to operate the system and exercise the specified test cases in order to implement the associated test design.‖

The test procedure or test script spec defines the step-by-step details of exactly how to perform the test cases. Here's the information that needs to be defined:

  - **Identifier.** A unique identifier that ties the test procedure to the associated test cases and test design.

- **Purpose.** The purpose of the procedure and reference to the test cases that it will exe-cute.
- **Special requirements**. Other procedures, special testing skills, or special equipment needed to run the procedure.
- **Procedure steps.** Detailed description of how the tests are to be run:
- **Log.** Tells how and by what method the results and observations will be recorded.
- **Setup.** Explains how to prepare for the test.
- **Start.** Explains the steps used to start the test.
- **Procedure.** Describes the steps used to run the tests.
- **Measure.** Describes how the results are to be determined for example, with a stopwatch or
- visual determination.
- **Shut down.** Explains the steps for suspending the test for unexpected reasons.
- **Restart.** Tells the tester how to pick up the test at a certain point if there's a failure or after shutting down.
- **Stop.** Describes the steps for an orderly halt to the test.
- **Wrap up.** Explains how to restore the environment to its pre-test condition.
- **Contingencies.** Explains what to do if things don't go as planned.

Q **What is test plan? List test planning activities.**

Ans: Like any project, the testing also should be driven by a plan. The test plan acts as the anchor for the execution, tracking and reporting of the entire testing project. Activities of test plan:

- Scope Management: Deciding what features to be tested and not to be tested.
- Deciding Test approach /strategy: Which type of testing shall be done like configuration, integration, localization etc.
- Setting up criteria for testing: There must be clear entry and exit criteria for different phases of testing. The test strategies for the various features and combinations determined how these features and combinations would be tested.
- 4.Identifying responsibilities, staffing and training needs
- Identifying resource requirements
- Identifying test deliverables
- Testing tasks: size and effort estimation

Q **How to prepare a test plan? Explain in detail.**

Ans: The test plan acts as the anchor for the execution, tracking and reporting of the entire testing project and covers.

1.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.
- Scope management: It entails:
    - Understanding what constitutes a release of product.
    - Breaking down the release into features.
    - Prioritizing the feature of testing.
- Deciding which features will be tested & which will not be

3.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?
- 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.
- Identifying responsibilities, staffing & Training needs: The next aspect of planning is who part of it. Identifying responsibilities , staffing & training needs addresses this aspect. 6.Identifying Resource Requirement: As a part of planning for a testing project, the project manager should provide estimate for the various h/w & s/w resources required.
- Identifying Test Deliverables: It includes : test plan itself, test case design specification, test cases , test logs & test summary report
- Testing task: Size and Effort estimation: This gives estimation in terms of size, effort & schedule of testing project.

Q **Describe test reporting in detail and how to prepare a summary report.**

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

1. Test incident report:

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

2. Test cycle report:

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

Test cycle report gives

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

3 Test summary report:

The final step in a test cycle is to recommend the suitability of a product for release. A report that summarizes the result of a test cycle is the test summary report.

There are two types of test summary report:

- Phase wise test summary ,which is produced at the end of every phase
- Final test summary report .
- A Summary report should present

1. Test Summary report Identifier
    - Description
        - Identify the test items being reported in this report with test id
    - Variances

- Mention any deviation from test plans, test procedures, if any.
- Summary of results
  - All the results are mentioned here with the resolved incidents and their solutions.
- Comprehensive assessment and recommendation for release should include

  Fit for release assessment and recommendation of release

Q **Describe test case specification of test process.**

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

Q **Describe Test Management Process and give details of following internal standards for process and method :**
(i) Naming and storage contention.
(ii) Documentation standard.

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

1. **Naming and storage conventions** for test artifacts: Every test artifacts(test specification, test case, test results and so on)have to be named appropriately and meaningfully. It enables
a) Easy identification of the product functionality.
b) Reverse mapping to identify the functionality corresponding to a given set of tests.
e.g. modules shall be M01,M02.Files types can be .sh, .SQL.
In addition to file naming conventions, the standards may also stipulate the conventions for directory structures for tests. These directory structures are mapped into configuration management repository.
2. **Documentation standards**: Documentation standards specify how to capture information about the tests within the test scripts themselves. It should include:
a. Appropriate header level comments at the beginning of a file that outlines the functions to be served by the test.
b. Sufficient inline comments, spread throughout the file
c. Up-to-Date change history information, reading all the changes made to the test file.

Q **How to identify resource requirement of test plan?**

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

Some of the following factors need to be considered:

1. Machine configuration (RAM,processor,disk)needed to run the product under test.
2. Overheads required by test automation tools, if any
3. Supporting tools such as compilers, test data generators, configuration management tools.
4. The different configurations of the supporting software(e.g. OS)that must be present

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

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

OR

Human Resource:
The following table represents various members in your project team
System Resource:
For testing, a web application, you should plan the resources as following tables:

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

Q How test case specifications useful in designing test cases?

Ans: The test case specifications should be developed from the test plan and are the second phase of the test development life cycle. The test specification should explain "how" to implement the test cases described in the test plan. Test case specifications are useful as it enlists the specification details of the items.
Test Specification Items are must for each test specification should contain the following items:
1. Case No.: The test case number should be a three digit identifier of the following form: c.s.t, where: c- is the chapter number, s- is the section number, and t- is the test case number.
2. Title: is the title of the test.
3. Programme: is the program name containing the test.

4. Author: is the person who wrote the test specification.
5. Date: is the date of the last revision to the test case.
6. Background: (Objectives, Assumptions, References, Success Criteria): Describes in words how to conduct the test.
7. Expected Error(s): Describes any errors expected
8. Reference(s): Lists reference documentation used to design the specification.
9. Data: (Tx Data, Predicted Rx Data): Describes the data flows between the Implementation under Test (IUT) and the test engine.
10. Script: (Pseudo Code for Coding Tests): Pseudo code (or real code) used to conduct the test.

Q **Write steps to prepare test plan. Also write features to be tested.0**

Ans: Features to be tested are:

1. Lists/tables: Files, features and functions, inputs and outputs, error messages
2. Outlines :E.g., function list, top-level/user-visible functions, sub-functions (options or submenus), entry and exit conditions on fully parameterized methods
3. Matrices: List function/operation vs. test conditions, e.g., the save operation with the following conditions: disk full, almost full, write-protected
4. Notes : How to run test, expected results, special instructions, one-shot or regression test, what test is looking for, assumptions in the test.

Q What is decision table? How to use decision tables for creating test-design for credit card example?

Ans:

- A **decision table** is a good way to deal with combinations of things (e.g. inputs). This technique is sometimes also referred to as a cause-effect' table. The first task is to identify a suitable function or subsystem which reacts according to a combination of inputs or events. The system should not contain too many inputs otherwise the number of combinations will become unmanageable. It is better to deal with large numbers of conditions by dividing them into subsets and dealing with the subsets one at a time. Once you have identified the aspects that need to be combined, then you put them into a table listing all the combinations of True and False for each of the aspects.

Conditions	Rule 1	Rule 2	Rule 3	Rule 4:
Pin Number	T	T	T	F
Payment Detail	T	F	F	T
Overdue details	F	T	T	F

Q Why is it essential to setup criteria for testing? List any three criteria in different situations.

Ans: There is a need to setup criteria for testing because:

1. An early start to testing reduces the cost, time to rework and error free software that is delivered to the client.

2. Software Development Life Cycle (SDLC) testing can be started from the Requirements Gathering phase and lasts till the deployment of the software.

3. It also depends on the development model that is being used. For example in Water fall model formal testing is conducted in the Testing phase, but in incremental model, testing is performed at the end of every increment/iteration and at the end the whole application is tested.

4. Testing is done in different forms at every phase of SDLC like during Requirement gathering phase, the analysis and verification of requirements are also considered testing.

5. Reviewing the design in the design phase with intent to improve the design is also considered as testing.

6. Testing performed by a developer on completion of the code is also categorized as Unit type of testing.

Any 3 criteria's in different situation are :

- Start with the static white box testing procedure when the specifications of the software to be developed are ready.

- Use the code coverage analyzer to test whether the whole code is getting executed and covered.

Perform unit testing as soon as one of the unit or sub module in the software is ready.

Q Which are features for selecting static test tools? Also list any two available test tools (static).

Ans: Features for selecting static test tools:
i. Assessment of the organization's maturity (e.g. readiness for change);
ii. Identification of the areas within the organization where tool support will help to improve testing processes;

iii. Evaluation of tools against clear requirements and objective criteria;
iv. Proof-of-concept to see whether the product works as desired and meets the requirements and objectives defined for it;
v. Evaluation of the vendor (training, support and other commercial aspects) or open-source network of support;
vi. Identifying and planning internal implementation (including coaching and mentoring for those new to the use of the tool).
Available static test tools are: 1. code coverage analyzer 2. Interface Analyzer

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

Ans: Test case is a well-documented procedure designed to test the functionality ofthe feature in the system.
For designing the test case, it needs to provide set of inputs and its corresponding expected outputs.
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.

5. Expected outputs which should be produced when there is no failure.

Q **Define software metrics. Describe product Vs process & objective Vs subjective metrics.**

Ans: 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.
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. 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.
- 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:

1. These metrics are a relatively new precept and are more flexible than the rigid framework of the objective metrics. Subjective metrics do deal with performance but the approach is more tailored. For some enterprises the niche in which they function forces project management to change in order to adapt to the demands of the workplace.

Q What is the use of code complexity testing? Also compute code complexity with the help of suitable example.

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

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

2. Branch Coverage (Code Complexity Testing)

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

3. Condition Coverage (Code Complexity Testing)

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

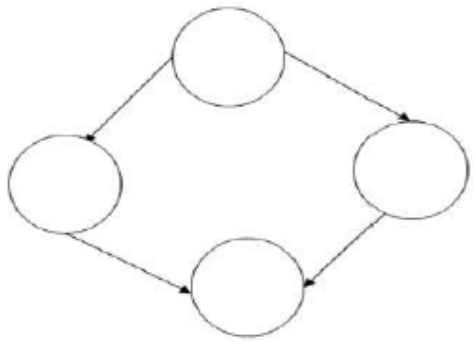
4. Code Complexity: Cyclomatic complexity is metric that quantifies the complexity of a program and provides answers to the questions such as

1. Which of the paths are independent? If two paths are not independent then we may be able to minimize the number of tests.

• 2. Is there an upper bound on the number of tests that must be run to ensure that all the statements have been executed at least once?
• In this a program is represented as flow graph. A flow graph consists of nodes and edges.
• Cyclo-matic complexity=Number of predicate nodes(P)+1

Or

• Cyclo-matic complexity=Edges(E)-Nodes(N)+2



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

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

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

Q What are types of test report? Write contents of test summary report.

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

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

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

Test cycle report gives :

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

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

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

A Summary report should content:

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

Q **Prepare six test cases for Admission form for college admission.**

Ans: Consider the college admission form having different fields such as Student‟s Name, Father‟s Name, Address,Phone,Caste,admission type,S.S.C percentage, SC Board, Submit button, Reset button.

Test Case Id	Test case Objectives	Input Data	Expected Result	Actual Result	Status
TC1	Name field	Any name(abcd xyz)	It should accept the name	The name is accepted	Pass
TC2	Phone Field	Any number having less than 10 digits(1234)	It should not accept. Should give error message “Please enter valid phone number”	Error message “Please enter valid phone number”	Pass
TC3	Phone Field	Any alphabets(abcde)	It should give error message as “Only Numbers”	Error message as “Only Numbers”	Pass
TC4	SSC Percentage Field	65	It should accept	It accepted	Pass
TC5	SSC Percentage Field	30	It should not accept. Should give error message.	Gives error message	Pass
TC6	Address field	Any characters(A-51, Market road, Mumbai)	It should accept.	It accepted	Pass

Q Describe how to identify responsibilities in testing.

Ans: A testing project requires different people to play different roles. There are roles of test engineers, test leads and test managers. There is also role definition on the dimensions of the modules being tested or the type of testing. These different roles should complement each other. The different role definition should.

12. Ensure there is clear accountability for a given task, so that each person knows what he or she has to do,
13. Clearly list the responsibilities for various functions to various people, so that everyone knows how his or her work fits into the entire project.
14. Complement each other, ensuring no one steps on an others‟ toes

4.Supplement each other, so that no task is left unassigned.

Role definition should not only address technical roles, but also list the management and reporting responsibilities. This includes frequency, format and recipients of status reports and other project-tracking mechanism.

Q Define metrics and measurements. Explain need of software measurement.

Ans: Metrics & measurement: Metrics is a relative measurement of status of process or product in terms of two or more entities taken together for comparison.
Measurements are key element for controlling software engineering processes.

Need of software measurements:

15. **Understanding**: Metrics can help in making the aspects of process more visible, thereby giving a better understanding of the relationship among the activities and entities they affect.
16. **Control**: Using baselines, goals and an understanding of the relationships, we can predict what is likely to happen and correspondingly, make appropriate changes in the process to help meet the goals.
17. **Improvement**: By taking corrective actions and making appropriate changes, we can improve a product. Similarly, based on the analysis of a project, a process can also be improved.

Q Write 4 test cases for user login form.

TC_Id	TC_name	Steps	Input data	Expected result	Actual Result	Status
TC_01	User name	Enter the username in alphanumeric alphabets A-Z Number 0-9	"abc123"	It should accept the username	It is accepted username	Pass
TC_02	Password	Enter the password in alphanumeric alphabets A-Z Number 0-9	"abc123"	It should accept the password	It is accepted password	pass
TC_03	Submit	1.After valid username and password 2. Click on submit button		It should goes to next page	It is going to next page	pass
TC_04	Cancel	Click on cancel button		It should remain in login page with blank fields	It shows login page with blank fields	pass

Q Describe two types of test reports.

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

1. **Test incident report:**

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

2. **Test cycle report:**

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

Test cycle report gives

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

3. **Test summary report:**

The final step in a test cycle is to recommend the suitability of a product for release. A report that summarizes the result of a test cycle is the test summary report.

There are two types of test summary report:

- Phase wise test summary ,which is produced at the end of every phase

2. Final test summary report.

A Summary report should present

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.

Q What factors shall be considered while selecting resource requirements?

1. **Ans:** Machine configuration (RAM, processor, disk, and so on) needed to run the product under test.
2. Overheads required by the test automation tool, if any
3. Supporting tools such as compilers , test data generators configuration management tools, and so on
4. The different configurations of the supporting software (for example, OS) that must be present
5. Special requirements for running machine- intensive tests such as load tests and performance tests
6. Appropriate number of license of all the software

OR

Factors to be considered while selecting the resource requirements are :

People: How many people are required?
How much experience they should posses? What kind of experience is needed?
What should they be expertise in?
Should they be full-time, part-time, contract, students?

Equipment: How many Computers are required?
What configuration computers will be required? What kind of test hardware is needed?
Any other devices like printers, tools etc.

Office and lab space: Where will they be located?
How big will they be?
How will they be arranged?

Software: Word processors, databases, custom tools. What will be purchased, what needs to be written?

Outsource companies: Will they be used? What criteria will be used for choosing them? How much will they cost?

Miscellaneous supplies: Disks, phones, reference books, training material. What else might be necessary over the course of the project? The specific resource requirements are very project-, team-, and company-dependent, so the test plan effort will need to

carefully evaluate what will be needed to test the software.

Q Explain three types of product metrics.

Ans: **Product Metrics is classified as :**

1. **Project Metrics**: A set of metrics that indicates how the project is planned and executed.
2. **Progress Metrics:** A set of metrics that tracks how the different activities of the project are progressing. It includes both development activities and testing activities. Progress Metrics is classified as a. Test defect metrics b. Development defect metrics
   - **Test defect metrics:** help the testing team in analysis of product quality and testing
   - **Development defect metrics:** help the development team in analysis of development activities.
3. **3. Productivity Metrics:** A set of metrics that takes into account various productivity
4. numbers that can be collected and used for planning and tracking testing activities. These
5. metrics help in planning and estimating of testing activities.

Q Explain any two internal standards in test management.

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:

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.

Q What are the things that test case specification shall identify?

Ans:

1. Test cases specify the inputs, predicted results and execution conditions. Each test case should aim to evaluate the operation of a key element or function of the system.

2. Failure of a test case, depending upon the severity of the failure, would be catalogued as part of the overall evaluation of the suitability of the system for its intended use.

3. Test cases can start with a specific _form‘ that allows operator entry of data into the system. This needs to be mapped, if the architecture is based upon an n-tier solution, through the business logic and rules into the server systems with transactions being evaluated both in a _nominal‘ mode where the transaction is a success and for those occasions when the transaction or _thread‘ fails.

4. Test design may also require one or more test cases and one or more test cases may be executed by a test procedure.

Defect Management

Q1) **Which parameters are considered while writing good defect report? Also write contents of defect template.**

Or

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

Ans: 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, testenvironment, 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

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.
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.
Fixed Build Version	Build version of the product where the defect was fixed (e.g. 1.2.3.9)

Q **Explain defect classification.**

Ans: A Software Defect / Bug is a condition in a software product which does not meet a software requirement (as stated in the requirement specifications) or end-user expectations (which may not be specified but are reasonable). In other words, a defect is an error in coding or logic that causes a program to malfunction or to produce incorrect/unexpected results.

•A program that contains a large number of bugs is said to be buggy.
•Reports detailing bugs in software are known as bug reports.
•Applications for tracking bugs are known as bug tracking tools.
•The process of finding the cause of bugs is known as debugging.
•The process of intentionally injecting bugs in a software program, to estimate test coverage by monitoring the detection of those bugs, is known as debugging.

CLASSIFICATION

Software Defects/ Bugs are normally classified as per:

•Severity / Impact
•Probability / Visibility
•Priority / Urgency

•Related Dimension of Quality
•Related Module / Component
•Phase Detected
•Phase Injected

OR

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 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 Defect: Testing defect are defects introduced in an application due to wrong testing , or defects in the test artifact leading to wrong testing. Defects which cannot be reproduced , or are not supported by requirement or are duplicate may represent a false call .In this defects includes

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.

Q **Explain defect management process with proper diagram.**

Ans:

Defect Management Process diagram:

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

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.

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.

Defect Resolution -- Work by the development team to prioritize, schedule and fix a defect, and document the resolution. This also includes notification back to the tester to ensure that the resolution is verified.

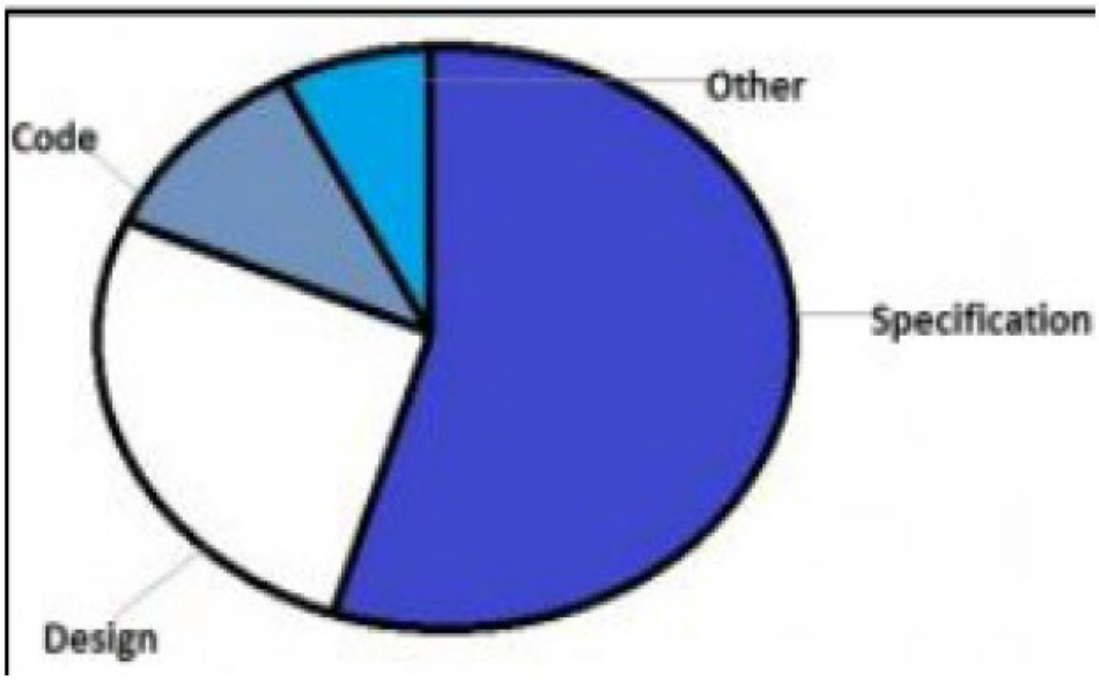
Process Improvement -- Identification and analysis of the process in which a defect originated to identify ways to improve the process to prevent future occurrences of similar defects. Also the validation process that should have identified the defect earlier is analyzed to determine ways to strengthen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

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

Q **Which are the different causes of software defects?**

Ans: 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:



Q **What are the points considered while estimating impact of a defect? Also explain techniques to find defect in short.**

Ans: Points to be considered while estimating the impacts of a defect are :
i. There is a strong relationship between the number of test cases and the number of function points.
ii. There is a strong relationship between the number of defects and the number of test cases and number of function points.
iii. The number of acceptance test cases can be estimated by multiplying the number of function points by 1.2.
iv. Acceptance test cases should be independent of technology and implementation techniques.
v. If a software project was 100 function points the estimated number of test cases would be 120.
vi. To estimate the number of potential defects is more involved.

Techniques to find defects are:
a) Quick Attacks: The quick-attacks technique allows you to perform a cursory analysis of a system in a very compressed timeframe. Even without a specification, you know a little bit about the software, so the time spent is also time invested in developing expertise. The skill is relatively easy to learn, and once you've attained some mastery your quick-attack session will probably produce a few bugs. Finally, quick attacks are quick. They can help you to make a rapid assessment. You may not know the requirements, but if your attacks yielded a lot of bugs, the programmers probably aren't thinking about exceptional conditions, and it's also likely that they made mistakes in the main functionality. If your attacks don't yield any defects, you may have some confidence in the general, happy-path functionality
b) Equivalence and Boundary Conditions: Boundaries and equivalence classes give us a technique to reduce an infinite test set into something manageable. They also provide a mechanism for us to show that the requirements are "covered".

c) Common Failure Modes: The heart of this method is to figure out what failures are common for the platform, the project, or the team; then try that test again on this build. If your team is new, or you haven't previously tracked bugs, you can still write down defects that "feel" recurring as they occur and start checking for them.
d) State-Transition Diagrams: Mapping out the application provides a list of immediate, powerful test ideas. Model can be improved by collaborating with the whole team to find "hidden" states transitions that might be known only by the original programmer or

specification author. Once you have the map, you can have other people draw their own diagrams, and then compare theirs to yours. The differences in those maps can indicate gaps in the requirements, defects in the software, or at least different expectations among team members. e) Use Cases and Soap Opera Tests: Use cases and scenarios focus on software in its role to enable a human being to do something. Use cases and scenarios tend to resonate with business customers, and if done as part of the requirement process, they sort of magically generate test cases from the requirements. They make sense and can provide a straightforward set of confirmatory tests. Soap opera tests offer more power, and they can combine many test types into one execution.

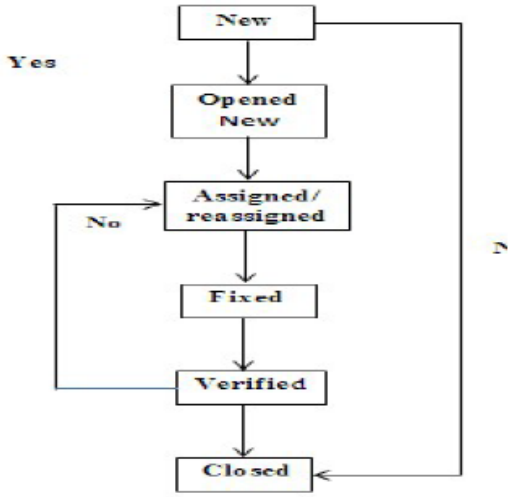
f) Code-Based Coverage Models: Imagine that you have a black-box recorder that writes down every single line of code as it executes. Programmers love code coverage. It allows them to attach a number an actual, hard, real number, such as 75% to the performance of their unit tests, and they can challenge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

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

Q **Explain defect life cycle to identify status of defect with proper labelled diagram.**

Ans:

Defect/Bug Life cycle:



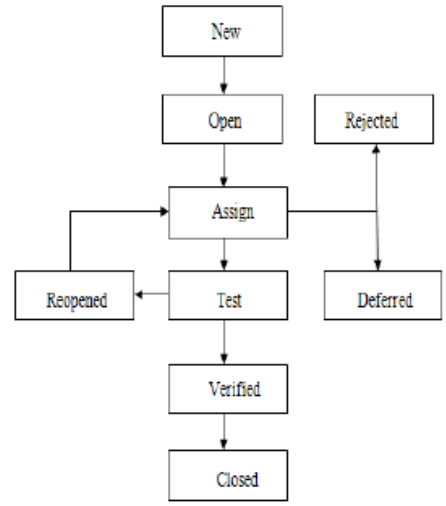


Fig: Defect life cycle

OR

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

2.Assigned: After the tester has posted the bug, the lead of the tester approves that the bugis genuine and he assigns the bug to corresponding developer and the developer team. It's 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.

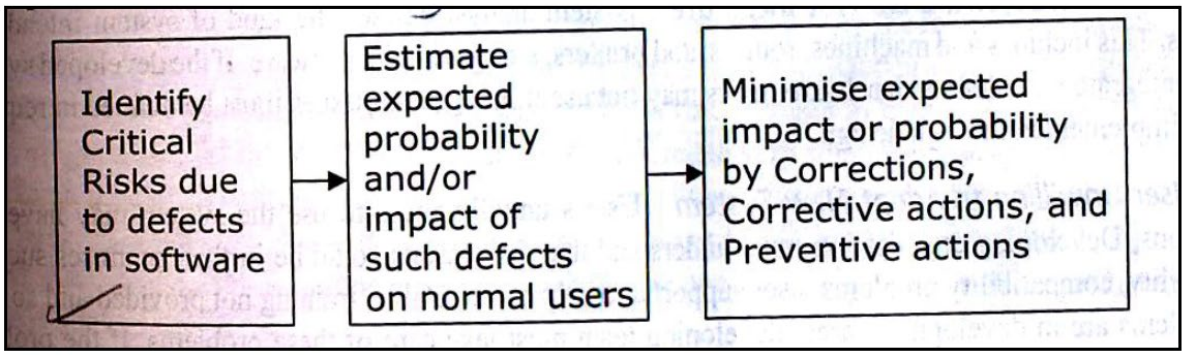
6.Retest: At this stage the tester do the retesting of the changed code which developer has

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.

Q **Illustrate defect prevention process of Defect fixing process with diagram.**

Ans: Defect prevention is a cycle of risk analysis and actions based on its ranking. Defects found in verification & validation must be realistic and represent the possible areas of improvement in development and testing processes. Defect must be governed by probability of happening & severity of the defect in terms of impact on the users and detection ability of occurrence of such defect prevention mechanism. The process of risk analysis & defect prevention is shown in fig.



Q **State how to minimize risk impact while estimating defect.**

Ans: Risk is a product of probability, impact and detection ability.
Risk minimization has three different methods of handling its probability, impact or detection ability. The decisions may be driven by organization policy, values, cost-benefit analysis etc. Minimization of problem due to risk happens in the following manner

- **Eliminate Risk**: Elimination of risk involves taking steps to remove risk from the root. Risk's probability is reduced to almost „O" by removing the causes of risk, so that the risk must not happen at all, or the organization user may be protected from the possible losses arising due to risk. Preventive controls can eliminate the probability of risk to a large extent. Preventive controls are management-

decided controls. Preventive controls are applied, if the probability of occurrence is very high. Preventive controls are useless, if the probability of happening is already negligible.

- **Mitigation of Risk**: Actions initiated by an organization to minimize the possible damage due to realization of risk are considered „mitigation actions". Mitigation actions are planned by an organization/project so that if the risk is realized, the impact due to it can be reduced to minimum possible. The corrective controls used from the mitigation action. Corrective controls maybe auto-corrective or suggestive.
- **Detection Ability Improvements**: Impact of a risk is more, if it catches the user unprepared. If people are aware of the risks, they can be well prepared to handle them. Generally, detective controls are used to increase the visibility towards risks. Sometimes, detective controls give threshold to corrective controls.
- **Contingency Planning**: Contingency planning refers to the actions initiated by an Organization, when preventive or corrective actions fail and risk actually occurs. They are Previously planned ways of tackling risks when all other planned activities for

Reducing Probability and impact of the risk fail, and the risk becomes reality.

Q **Explain the defect tracking with defect life cycle diagram and the different defect states.**

Ans:

Defect's (bug's) life cycle is as shown in the diagram below:

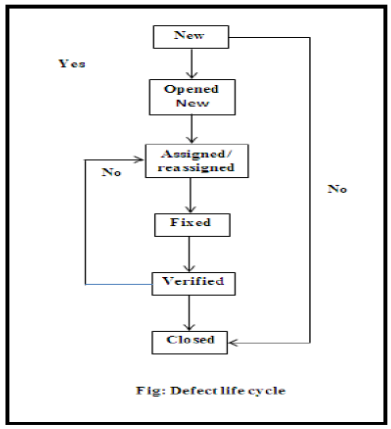
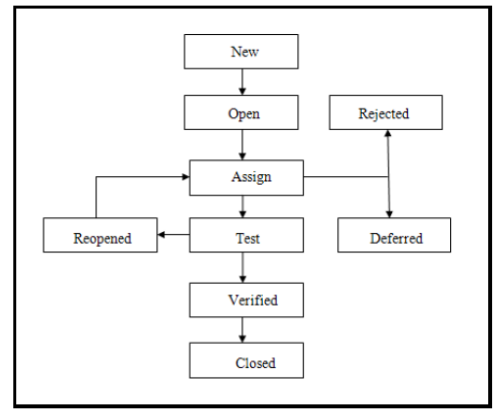


Fig: Defect life cycle

OR



- **Different defect states and their meanings are as shown below:**

Status	Alternative Status
NEW	
ASSIGNED	OPEN
DEFERRED	
DROPPED	REJECTED
COMPLETED	FIXED, RESOLVED, TEST
REASSIGNED	REOPENED
CLOSED	VERIFIED

The defect states are as explained below:

- **NEW:** Tester finds a defect and posts it with the status NEW. This defect is yet to be studied/approved. The fate of a NEW defect is one of ASSIGNED, DROPPED and DEFERRED.

- **ASSIGNED / OPEN:** Test / Development / Project lead studies the NEW defect and if it is found to be valid it is assigned to a member of the Development Team. The assigned Developer's responsibility is now to fix the defect and have it COMPLETED. Sometimes, ASSIGNED and OPEN can be different statuses. In that case, a defect can be open yet unassigned.
- **DEFERRED:** If a valid NEW or ASSIGNED defect is decided to be fixed in upcoming releases instead of the current release it is DEFERRED. This defect is ASSIGNED when the time comes.
- **DROPPED / REJECTED:** Test / Development/ Project lead studies the NEWdefect and if it is found to be invalid, it is DROPPED / REJECTED. Note that the specific reason for this action needs to be given.
- **COMPLETED / FIXED / RESOLVED / TEST:** Developer 'fixes' the defect that is ASSIGNED to him or her. Now, the 'fixed' defect needs to be verified by the Test Team and the Development Team 'assigns' the defect back to the Test Team. A COMPLETED defect is either CLOSED, if fine, or REASSIGNED, if still not fine.
- If a Developer cannot fix a defect, some organizations may offer the following statuses:
- **Won't Fix / Can't Fix:** The Developer will not or cannot fix the defect due to some reason.
- **Can't Reproduce:** The Developer is unable to reproduce the defect.
- **Need More Information:** The Developer needs more information on the defect from the Tester.
- **REASSIGNED / REOPENED:** If the Tester finds that the 'fixed' defect is in fact not fixed or only partially fixed, it is reassigned to the Developer who 'fixed' it.
  A REASSIGNED defect needs to be COMPLETED again.
- **CLOSED / VERIFIED:** If the Tester / Test Lead finds that the defect is indeed fixed and is no more of any concern, it is CLOSED / VERIFIED.

Q **What do you mean 'Defect Impact'? Explain how to estimate the defect impact.**

Ans: Defect Impact is a classification of software defect (bug) to indicate the degree of negative impact on the quality of software.

Or

Defect Impact: The degree of severity that a defect has on the development or operation of a component or system.

How to Estimate the defect impact

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 the risk becoming a problem and
I= Impact in dollars if the risk becomes a problem.

Once the expected impact of each risk is identified, the risks should be prioritized by the expected impact and the degree to which the expected impact can be reduced. While guess work will constitute a major role in producing these numbers, precision is not important. What will be important is to identify the risk, and determine the risk's order of magnitude.

Large, complex systems will have many critical risks. Whatever can be done to reduce the probability of each individual critical risk becoming a problem to a very small number should be done. Doing this increases the probability of a successful project by increasing the probability
that none of the critical risks will become a problem.
One should assume that an individual critical risk has a low probability of becoming a problem only when there is specific knowledge justifying why it is low. For example, the likelihood that an important requirement was missed may be high if developers have not involved users in the project. If users have actively participated in the requirements definition, and the new system is not a radical departure from an existing system or process, the likelihood may be low.

For example:

- An organization with a project of 2,500 function points and was about medium at defect discovery and removal would have 1,650 defects remaining after all defect removal and discovery activities.
- The calculation is 2,500 x 1.2 = 3,000 potential defects.
- The organization would be able to remove about 45% of the defects or 1,350 defects.
- The total potential defects (3,000) less the removed defects (1,350) equals the remaining defects of 1,650.

Q **The student passing the diploma shall be awarded class of passing as – Distinction, Fisrt Class, Second Class, Pass Class. The class will be applicable only if the student has his/her result 'pass' in all of subjects. Write the test cases (minimum-4) for the class awarding algorithm (module).**

Ans:

Step	Test step	Test data	Expected output	Actual output	Status
1	Enter the marks in the range of 40-49%	Any value between 40 to 49	Should display class as Pass class	Displayed the class as Pass class	Pass
2	Enter the marks in the range of 50-59%	Any value between 50-59	Should display as Second class	Displayed Second class	Pass
3	Enter marks in the range of 60-74%	Any value between 60-74	Should display as first class	Display as First class	Pass
4	Enter the marks in the range of 75 to 100	Any value between 75-100	Should display as Distinction	Displayed the result as Distinction	Pass

Q Describe the requirement defects and coding defects in details.

Ans: **Requirements defects**: A valid requirement which is mandatory and supposed to code (or implement) is missed. The root cause of this defect is due to not capturing this requirement in the specification document. These types of observations are categorized as _Missed requirements defects' since the requirements are missed from requirement specification document.

These types of defects are usually caused by business analyst's oversight.

Example:

Tester observation while testing the Website: Tester found that _Disclaimer' link is missing in the website. According to organization/webmaster guidelines it is mandatory to show _Disclaimer' link in the website so tester expressed his/her concern that the _Disclaimer' link is missing and to fix this developer expects the business analyst to document in requirement specification document.

Coding defects: The requirements have been coded incorrectly due to which behavior of an implemented software function is not in accordance with the requirement specification documents.

The other frequently occurred defects made by developers are due to

Missed to code for the requirements which listed in requirement specification document Coding the requirements which are not specified in the requirement specification document

These types of defects are usually caused by developer's oversight.

Example:

Requirement as per specification document: If user clicks on _Home' link in a website then _Home' page should be presented to the user.

Tester observation while testing the _Home' link: Tester found that _About Me' page is displayed each time the _Home' link is clicked which is a deviation in the behavior from the requirement specification document. This is an example of coding defect.

Q **Write four test cases to test sign-in form of gmail account.**
Ans:

Test Case Id	Description	Input Data	Expected Result	Actual Result	Status
TC1	Login(email id) Field of Sign-in form of Gmail	Enter –abc123‖ and click on –Next‖ button	It shall prompt to enter Password	It prompts to enter password	Pass
TC2	Password field of Sign-in form of Gmail	Enter –xyz‖ (valid id)and click on –Sign in‖ button	It shall open Gmail account	It opens Gmail account	Pass
TC3	Login(email id) Field of Sign-in form of Gmail	Without entering login id, click on "Next" button	It shall give message as –Please enter your email"	It gives message as –Please enter your email"	Pass
TC4	Password field of Sign-in form of Gmail	Without entering password, click on "Sign in" button	It shall give message as "Please enter your password"	It gives message as "Please enter your password"	Pass
TC5	Login(email id) Field of Sign-in form of Gmail	Enter "pqr" (invalid id)and click on "Next button"	It shall give message as "Sorry, Google doesn't recognize that email"	It give message as" Sorry, Google doesn't recognize that email"	Pass
TC6	Password field of Sign-in form of Gmail	Enter "abc123" (invalid password) at password field after entering valid login id.	It shall give message as "Wrong password. Try again"	It gives message "Wrong password. Try again"	Pass

Q Write the entity, purpose and attributes of the following elements of test infrastructure management.
i) A test case database(TCDB)
ii) A defect repository

Entity	Purpose	Attributes
Test case	Records all the "static" information about the tests	• Test case ID • Test case name (filename) • Test case owner • Associated files for the test case
Test case- product cross-reference	Provides a mapping between the tests and the corresponding product features ; enables identification of tests for a given feature	• Test case ID • Modulate ID
Test case run history	Gives the history of when a test was run and what was the result; provides inputs on selection of tests for regression runs (see chapter 8)	• Test case ID • Run date • Time taken • Run status (success/failure)
Test case – Defect cross-reference	Gives details of test cases introduced to test certain specific defects detected in the product ;provides inputs on the selection of tests for regression runs	• Test case ID • Defect reference# (points to a record in the defect repository)

ii). Defect Repository:

Entity	Purpose	Attributes
Defect details	Records all the " static" information about the tests	• Defect ID • Defect priority /severity • Defect description • Affected product(s) • Any relevant version information (for example, OS version) • Customers who encountered the problem (could be reported by the internal testing team also) • Date and time of defect occurrence
Defect test details	Provides details of test cases for a given defect. Cross-references the TCDB	• Defect ID • Test case ID
Fix details	Provides details of fixes for a given defect; cross-references the configuration management repository	• Defect ID • Fix details (file changed ,fix release information)
Communication	Captures all the details of the communication that transpired for this defect among the various stakeholders these could include communication between the testing team and development team, customer communication, and so on. Provides insights into effectiveness of communication	• • Test case ID • Defect reference # • Details of communication

Q **What are the different points to be noted in reporting defects?**

Ans: It is essential that you report defects effectively so that time and effort is not unnecessarily wasted in trying to understand and reproduce the defect. Here are some guidelines:

i. Be specific:

- Specify the exact action: Do not say something like _Select Button B'.
- Do you mean _Click Button B' or _Press ALT+B' or _Focus on Button B and click ENTER'.
- In case of multiple paths, mention the exact path you followed: Do not say something like —If you do _A and X' or _B and Y' or _C and Z', you get D.‖ Understanding all the paths at once will be difficult. Instead, say —Do _A and X' and you get D.‖ You can, of course, mention elsewhere in the report that —D can also be got if you do _B and Y' or _C and Z'.‖
- Do not use vague pronouns: Do not say something like —In Application A, open X, Y, and Z, and then close it.‖ What does the _it' stand for? _Z' or, _Y', or _X' or _Application A'?‖

ii. Be detailed:

- Provide more information (not less). In other words, do not be lazy.
- Developers may or may not use all the information you provide but they sure do not want to beg you for any information you have missed.

iii. Be objective:

- Do not make subjective statements like —This is a lousy application‖ or —You fixed it real bad.‖
- Stick to the facts and avoid the emotions.

iv. Reproduce the defect:

- Do not be impatient and file a defect report as soon as you uncover a defect. Replicate it at least once more to be sure.

v. Review the report:

- Do not hit _Submit‘ as soon as you write the report.
- Review it at least once.
- Remove any typing errors.

Testing Tools and Measurements

Q1) **Which different benefits help to recommend automated testing? Write advantages of switching to automated testing.**

Ans: NEED of automated testing

i. An automated testing tool is able to playback pre-recorded and predefined actions, compare the results to the expected behavior and report the success or failure of these manual tests to a test engineer.

ii. Once automated tests are created they can easily be repeated and they can be extended to perform tasks impossible with manual testing.

iii. Because of this, savvy managers have found that automated software testing is an essential component of successful development projects.

1. Automated Software Testing Saves Time and Money

i. Software tests have to be repeated often during development cycles to ensure quality. Every time source code is modified software tests should be repeated.

ii. For each release of the software it may be tested on all supported operating systems and hardware configurations.

iii. Manually repeating these tests is costly and time consuming. Once created, automated tests can be run over and over again at no additional cost and they are much faster than manual tests.

iv. Automated software testing can reduce the time to run repetitive tests from days to hours.

v. A time savings that translates directly into cost savings.

2. Testing Improves Accuracy

i. Even the most conscientious tester will make mistakes during monotonous manual testing.

ii. Automated tests perform the same steps precisely every time they are executed and never forget to record detailed results.

3. Increase Test Coverage

i. Automated software testing can increase the depth and scope of tests to help improve software quality.

ii. Lengthy tests that are often avoided during manual testing can be run unattended.

iii. They can even be run on multiple computers with different configurations.

iv. Automated software testing can look inside an application and see memory contents, data tables, file contents, and internal program states to determine if the product is behaving as expected.

v. Automated software tests can easily execute thousands of different complex test cases during every test run providing coverage that is impossible with manual tests.

vi. Testers freed from repetitive manual tests have more time to create new automated software tests and deal with complex features.

Q) **State limitations of manual testing. Write any four.**

Ans: 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. GUI 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.
Batch testing is not possible, for each and every test execution Human user interaction is mandatory.
viii. Comparing large amount of data is impractical.
ix. Processing change requests during software maintenance takes more time.

Q **Enlist factors considered for selecting a testing tool for test automation.**

Ans: Criteria for Selecting Test Tools:
The Criteria's 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 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.

Q **State & explain any four benefits of automation in testing.**

OR
Elaborate the advantages (any four) of using the test automation tools.

Ans: Benefits of automation testing:
1. Speed: Think about how long it would take you to manually try a few thousand test cases for the windows Calculator. You might average a test case every five seconds or so. Automation might be able to run 10, 100 even 1000 times that fast.
2. Efficiency: While you are busy running test cases, you can't be doing anything else. If you have a test tool that reduces the time it takes for you to run your tests, you have more time for test planning and thinking up new tests.
3. Accuracy and Precision: After trying a few hundred cases, your attention may reduce and you will start to make mistakes .A test tool will perform the same test and check the result perfectly, each and every time.
4. Resource Reduction: Sometimes it can be physically impossible to perform a certain test case. The number of people or the amount of equipment required to create the test condition could be prohibitive. A test tool can used to simulate the real world and greatly reduce the physical resources necessary to perform the testing.
5. Simulation and Emulation: Test tools are used to replace hardware or software that would normally interface to your product. This "face" device or application can then be used to drive or respond to your software in ways that you choose-and ways that might otherwise be difficult to achieve.
6. Relentlessness: Test tool and automation never tire or give up. It will continuously test the software.

OR

Benefits of Automation Testing are:
1. **Save Time /Speed**: Due to advanced computing facilities, automation test tools prevail in speed of processing the tests. Automation saves time as software can execute test cases faster than human.
2. **Reduces the tester's involvement in executing tests**: It relieves the testers to do some other work.
3. Repeatability/Consistency: The same tests can be re-run in exactly the same manner eliminating the risk of human errors such as testers forgetting their exact actions, intentionally omitting steps from the test scripts, missing out steps from the test script, all of which can result in either defects not being identified or the reporting of invalid bugs (which can again, be time consuming for both developers and testers to reproduce)
4. Simulated Testing: Automated tools can create many concurrent virtual users/data and effectively test the project in the test environment before releasing the product.
5. Test case design: Automated tools can be used to design test cases also.
Through automation, better coverage can be guaranteed than if done manually.
6. Reusable: The automated tests can be reused on different versions of the software, even if the interface changes.
7. **Avoids human mistakes:** Manually executing the test cases may incorporate errors. But this can be avoided in automation testing.
8. **Internal Testing:** Testing may require testing for memory leakage or checking the coverage of testing. Automation can done this easily.
9. Cost Reduction: If testing time increases cost of the software also increases. Due to testing tools time and therefore cost is reduced.

Q Differentiate between manual testing & automation testing.

Ans:

Automated Testing	Manual Testing
• If you have to run a set of tests repeatedly automation is a huge gain	• If Test Cases have to be run a small number of times it's more likely to perform manual testing
• Helps performing "compatibility testing" - testing the software on different configurations	• It allows the tester to perform more ad-hoc (random testing)
• It gives you the ability to run automation scenarios to perform regressions in a shorter time	• Short term costs are reduced
• It gives you the ability to run regressions on a code that is continuously changing	• The more time tester spends testing a module the grater the odds to find real user bugs
• It's more expensive to automate. Initial investments are bigger than manual testing	• Manual tests can be very time consuming
• You cannot automate everything, some tests still have to be done manually	• For every release you must rerun the same set of tests which can be tiresome

Q **How to select a testing tool? Explain in detail.**

Ans: **Criteria for Selecting Test Tools:**
The Categories for selecting Test Tools are,

- Meeting requirements;
- Technology expectations;
- Training/skills;
- Management aspects.

1. **Meeting requirements-**

There are plenty of tools available in the market but rarely do they meet allthe 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 hardware and software to be upgraded. This increases the cost of the already- expensive test tool.

OR

Guidelines for selecting a tool:

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

- Select affordable tools. Cost and benefits of various tools must be compared before making final decision.
- Backdoor entry of tools must be prevented. Unauthorized entry results into failure of tool and creates a negative environment for new tool introduction.

Q **Which types of test are first candidates for test automation? Why?**

Ans: **Stress, reliability, scalability and performance testing:** These types of testing require the test case to be run from a large number of different machines for an extended period of time, such as 24 hours, 48 hours, and so on. It is just not possible to have hundreds of users trying out the product they may be not willing to perform the repetitive tasks, nor will it be possible to find that many people with the required skill sets. Test cases belonging to these testing types become the first candidates for automation.

Regression tests: Regression tests are repetitive in nature .These test cases are executed multiple times during the product development phase. Given the repetitive nature of test cases, automation will save significant time and effort in the long run. The time thus gained can be effectively utilized for other tests.

Functional tests: These kinds of tests may require a complex set up and thus require specialized skill, which may not be available on an ongoing basis. Automating these once, using the expert skill sets, can enable using less-skilled people to run these test on an ongoing basis.

Q **List what are the different guidelines to be followed while selecting dynamic test tools.**

Ans: i) Assessment of the organization‘s maturity (e.g. readiness for change);

- Identification of the areas within the organization where tool support will help to improve testing processes;
- Evaluation of tools against clear requirements and objective criteria;
- Proof-of-concept to see whether the product works as desired and meets the requirements and objectives defined for it;
- Evaluation of the vendor (training, support and other commercial aspects) or open-source network of support;
- Identifying and planning internal implementation (including coaching and mentoring for those new to the use of the tool).