MODUEL :-2

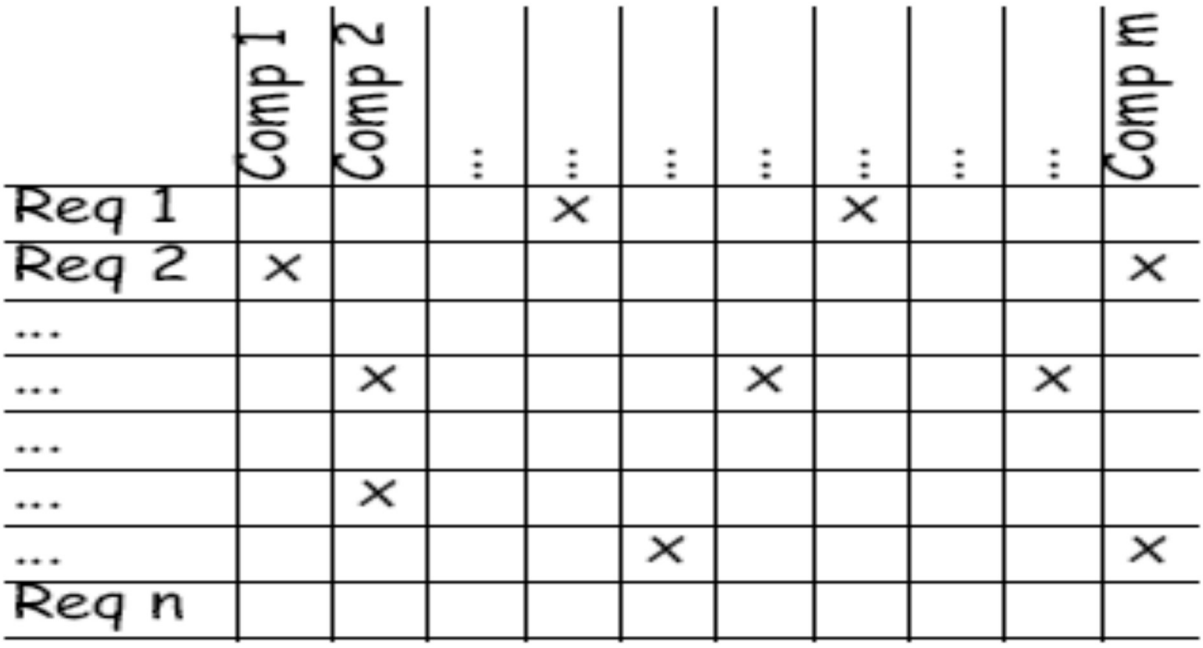
(Q.1) What is Exploratory Testing?

- In exploratory testing tester focuses more on how the software actually works, testers do minimum planning and maximum execution of the software by which they get in depth idea about the software functionality, once the tester starts getting insight into the software, he can make decisions to what to test next. Exploratory testing is mostly used if the requirements are incomplete and time to release the software is less.

    - Test design, execution and logging happen simultaneously.
    - Testing is often not recorded.
    - Makes use of experience and test patterns.
    - Though the current trend in testing is to push for automation, exploratory testing is a new way of thinking automation has its limits.
    - Is not random testing but it is ad hoc testing with purpose of find bugs.
    - Is structured and rigorous.
    - Is cognitively (thinking) structured as compared to procedure structure of scripted testing.
    - This structure comes from charter time boxing etc...
    - Its highly teachable and manageable.

(Q.2) What is traceability matrix?

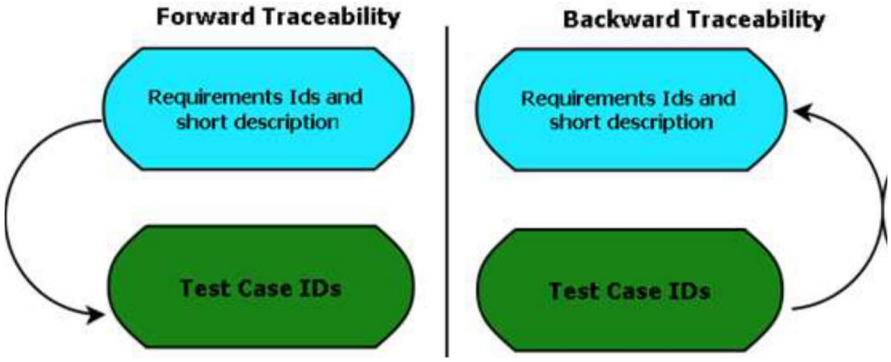
- Test conditions should be able to be linked back to their sources in the test basis, this is known as traceability.
- To protect against changes, you should be able to trace back from every system component to the original requirement that caused its presence.
- A software process should help you keeping the virtual table up to date.



MODUEL :-2

Types of Traceability Matrix

- Forward Traceability – Mapping of Requirements to Test cases
- Backward Traceability – Mapping of Test Cases to Requirements



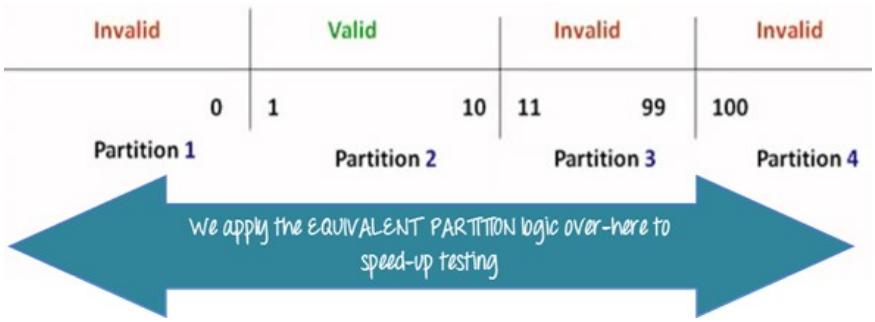
(Q.3) What is Boundary value testing?

- Boundary values analysis is a methodology for designing test cases that concentrates software testing effort on cases near the limits of valid ranges.
- Boundary value analysis is a method which refines equivalence partitioning.
- Boundary value analysis uses some analysis of partition as EP and is usually used in conjunction with EP in test case design.

(Q.4) What is Equivalence partitioning testing?

- If a specific value is given as input, then one valid and two invalid equivalence classes are defined.
- If we want to test the following IF statement: if value is between 1 and 100 (inclusive) (e.g., value >=1 and <=100) then….
- Aim is to treat of inputs as equivalent and to select are representative input to test them all.
- EP can be used for all levels of testing.
- EP says that by testing just one value we have tested the partitions. (Typically, a mid point value is used) it assumes that
    - If one value funds a bug, the others probably will too.

- If one doesn't find a bug the others probably won't either.



- The valid partition is bounded by the values 1 and 100.
- Plus, there are 2 invalid partitions.

(Q.5) What is Integration testing?

- It verifies the proper executions of software components and proper interfacing between components with in the solution.
- Testing performed to expose defects in the interfaces and in the interactions between integrated components or systems.
- Integration testing is a level if the software testing process where individual units are Combined and tested as a group.

There are 2 levels of integration testing.

(1) component integration testing
(2) system integration testing

1. Component integration testing

- Testing performed to expose defects in the interface and interaction between integrated components.

2. System integration testing

- It tests the interactions between different systems and may be done after system testing.
- It verifies the proper executions of software components and proper interfacing between components with in the solution.

(Q.6) What determines the level of risk?

- Risk – 'A factor that could result in future negative consequences; usually expressed as impact and likelihood'

Types of Risk

- **Risks are of two types**

- Project Risks
- Product Risk

Types of Risk Examples

- Example of Project risk is Senior Team Member leaving the project abruptly.
- But just identifying the risk is not enough. You need to identify mitigation. In this case mitigation could be Knowledge Transfer to other team members & having a buffer tester in place
- Example of product risks would be Flight Reservation system not installing in test environment

- Mitigation in this case would be conducting a smoke or sanity testing. Accordingly you will make changes in your scope items to include sanity testing

(Q.7) what is alpha testing?

- It is always performed by the developers at the software development site.
- Sometimes it is also performed by Independent Testing Team.
- Alpha Testing is not open to the market and public
- It is conducted for the software application and project.
- It is always performed in Virtual Environment.
- It is always performed within the organization.
- It is the form of Acceptance Testing.
- Alpha Testing is definitely performed and carried out at the developing organizations location with the involvement of developers.
- It comes under the category of both White Box Testing and Black Box Testing.

(Q.8) what is beta testing?

- It is always performed by the customers at their own site.
- It is not performed by Independent Testing Team.
- Beta Testing is always open to the market and public.
- It is usually conducted for software product.
- It is performed in Real Time Environment.
- It is always performed outside the organization.
- It is also the form of Acceptance Testing.
- Beta Testing (field testing) is performed and carried out by users or you can say people At their own locations and site using customer data.
- It is only a kind of Black Box Testing.
- Beta testing can be considered **"**pre-release testing.
- Pilot Testing is testing to product on real world as well as collect data on the use of Product in the classroom.

(Q.9) What is component testing?

- Unit testing is a level of the software testing process where individual units/components of a software /system are tested. The purpose is to validate that each unit of the software performs as designed.
- A minimal software item that can be tested in isolated. It means A unit is the smallest testable part of software.
- The testing of individual software components.
- Unit testing is the first level of testing and is perform prior to integration testing.
- Sometimes known as unit testing module testing or program testing.

- Unit testing is performed by using the white box testing method.

  Beloved we look at some of what extreme programming brings to the world of unit testing.

  - Test are written before the code.
  - Rely heavily on testing from works
  - All classes in the applications are tested.
  - Quick and easy integration is mode possible.

(Q.10) What is functional system testing?

- A requirement that specifies a function that a system or system component must perform.

  ### There are two types of test approach.

  (1) Requirement based functional testing.
  (2) Process based testing.

  ### 1 Requirement based testing

  - Test procedures and cases derived from.
  - Details user requirements.
  - System requirements functional specification.
  - User documentation/instructions.
  - High level system design.

  ### 2 Business process-based testing

  Test procedures and cases derived from:

  - Expected user profiles
  - Business scenarios.
  - Use cases.

(Q.11) What is Non-Functional Testing?

- Testing the attributes of a component or system that do not relate to functionality, e.g., reliability, efficiency, usability, interoperability, maintainability and portability.
- May be performed at all Test levels (not just Non-Functional Systems Testing)
- Measuring the characteristics of the system/software that can be quantified on a varying scale- e.g., performance test scaling.

- Non-functional testing includes, but is not limited to, performance testing, load testing, stress testing, usability testing, maintainability testing, reliability testing and portability testing.
- It is the testing of "how" the system works. Non-functional testing may be performed at all test levels.
- The term non-functional testing describes the tests required to measure characteristics of systems and software that can be quantified on a varying scale, such as response times for performance testing.
- To address this issue, performance testing is carried out to check & fine tune system response times.
- The goal of performance testing is to reduce response time to an acceptable level.
- Hence load testing is carried out to check systems performance at different loads i.e., number of users accessing the system.

(Q.12) What is GUI Testing?

- Graphical User Interface (GUI) testing is the process of testing the system's GUI of the System under Test. GUI testing involves checking the screens with the controls like menus, buttons, icons, and all types of bars – tool bar, menu bar, dialog boxes and windows etc.

WHAT DO YOU CHECK IN GUI TESTING?

- Check all the GUI elements for size, position, width, length and acceptance of characters or numbers. For instance, you must be able to provide inputs to the input fields.
- Check you can execute the intended functionality of the application using the GUI
- Check Error Messages are displayed correctly
- Check for Clear demarcation of different sections on screen
- Check Font used in application is readable
- Check the alignment of the text is proper
- Check the Colour of the font and warning messages is aesthetically pleasing
- Check that the images have good clarity
- Check that the images are properly aligned
- Check the positioning of GUI elements for different screen resolution.

Approach of GUI Testing

- **MANUAL BASED TESTING**

- Under this approach, graphical screens are checked manually by testers in conformance

with the requirements stated in business requirements document.

- **RECORD AND REPLAY**

- GUI testing can be done using automation tools. This is done in 2 parts. During Record, test steps are captured into the automation tool. During playback, the recorded test steps are executed on the Application under Test. Example of such tools - QTP.

- **MODEL BASED TESTING**

- A model is a graphical description of system's behaviour. It helps us to understand and predict the system behaviour. Models help in a generation of efficient test cases using the system requirements.

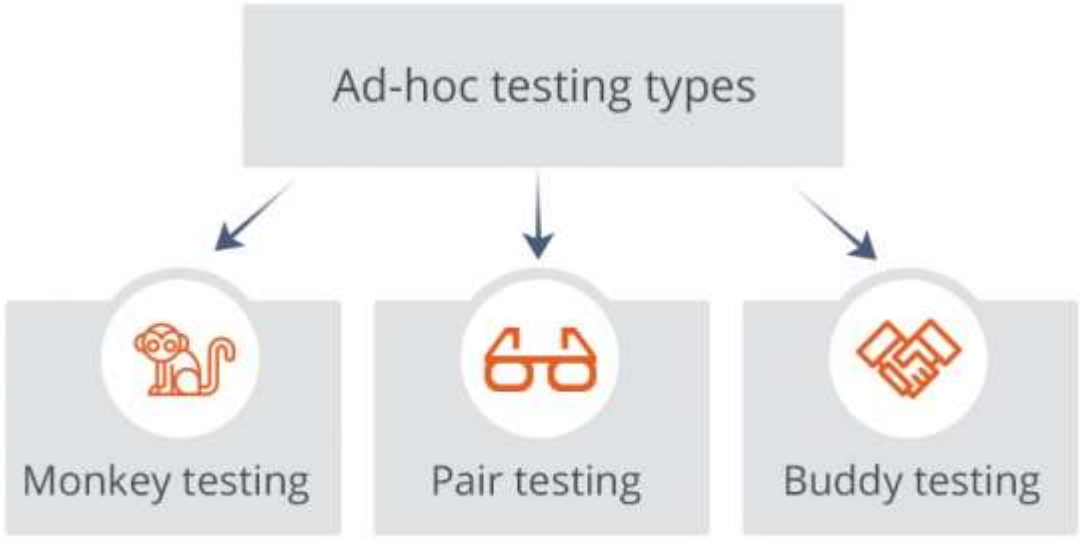
(Q.13) What is Adhoc testing?

- Adhoc testing is an informal testing type with an aim to break the system.
- It does not follow any test design techniques to create test cases.
- In fact, is does not create test cases altogether!
- This testing is primarily performed if the knowledge of testers in the system under test is very high.
- Main aim of this testing is to find defects by random checking.
- Adhoc testing can be achieved with the testing technique called Error Guessing.
- The Error guessing is a technique where the experienced and good testers are encouraged to think of situations in which the software may not be able to cope.
- Some people seem to be naturally good at testing and others are good testers because they have a lot of experience either as a tester or working with a particular system and so are able to find out its weaknesses.

There are different types of Adhoc testing and they are listed as below:

- **Buddy Testing**
  - Two buddies mutually work on identifying defects in the same module. Mostly one buddy will be from development team and another person will be from testing team.
- **Pair testing**
  - Two testers are assigned modules, share ideas and work on the same machines to find defects. roles of the parsons can be a tester and scriber during testing.
- **Monkey Testing**
  - Randomly test the product or application without test cases with a goal to break the system.

MODUEL :-2



(Q.14) What is load testing?

- **Load testing** - It's a performance testing to check system behaviour under load. Testing an application under heavy loads, such as testing of a web site under a range of loads to determine at what point the system's response time degrades or fails.
- This testing usually identifies –
  - The maximum operating capacity of an application
  - Determine whether current infrastructure is sufficient to run the application
  - Sustainability of application with respect to peak user load
  - Number of concurrent users that an application can support, and scalability to allow more users to access it.
  - It is a type of non-functional testing. Load testing is commonly used for the Client/Server, Web based applications – both Intranet and Internet.

(Q.15) What is stress Testing?

- **Stress testing** - System is stressed beyond its specifications to check how and when it fails. Performed under heavy load like putting large number beyond storage capacity, complex database queries, continuous input to system or database load.

(Q.16) What is white box testing and list the types of white box testing?

White Box Testing:

- Testing based on an analysis of the internal structure of the component or system.

MODUEL :-2

- Structure-based testing technique is also known as ‘white-box’ or ‘glass-box’ testing technique because here the testers require knowledge of how the software is implemented, how it works.
- White box testing is also called glass testing or open box testing. In order to perform white box testing on an application, the tester needs to possess knowledge of the internal working of the code.
- The tester needs to have a look inside the source code and find out which unit/chunk of the code is behaving inappropriately.

- Test coverage measures the amount of testing performed by a set of test. Wherever we can count things and can tell whether or not each of those things has been tested by some test, then we can measure coverage and is known as test coverage.

- The basic coverage measure is where the ‘coverage item’ is whatever we have been able to count and see whether a test has exercised or used this item.

$$\text{Coverage} = \frac{\text{Number of coverage items exercised}}{\text{Total number of coverage items}} \times 100\%$$

The different types of coverage are:

- Statement coverage
- Decision coverage
- Condition coverage

Statement/Segment Coverage:-

- The statement coverage is also known as line coverage or segment coverage.
- The statement coverage covers only the true conditions.

The statement coverage can be calculated as shown below:

$$\text{Statement coverage} = \frac{\text{Number of statements exercised}}{\text{Total number of statements}} \times 100\%$$

ADVANTAGE:

- It verifies what the written code is expected to do and not to do
- It measures the quality of code written
- It checks the flow of different paths in the program and it also ensure that whether
- those paths are tested or not.

DISADVANTAGE:

- It cannot test the false conditions.
- It does not report that whether the loop reaches its termination condition.
- It does not understand the logical operators.

Decision/Branch Coverage

- Decision coverage also known as branch coverage or all-edges coverage.
- It covers both the true and false conditions unlikely the statement coverage.
- A branch is the outcome of a decision, so branch coverage simply measures which decision outcomes have been tested.
- Aim is to demonstrate that all Decisions have been run at least once Decision and Branch Test coverage for a piece of code is often the same, but not always With an IF statement, the exit can either be TRUE or FALSE, depending on the value of the logical condition that comes after IF.

The decision coverage can be calculated as shown below:

$$\text{Decision coverage} = \frac{\text{Number of decision outcomes exercised}}{\text{Total number of decision outcomes}} \times 100\%$$

ADVANTAGES:

- To validate that all the branches in the code are reached
- To ensure that no branches lead to any abnormality of the program's operation
- It eliminates problems that occur with statement coverage testing

DISADVANTAGES:

- This metric ignores branches within Boolean expressions which occur due to short-circuit operators.

NOTE:

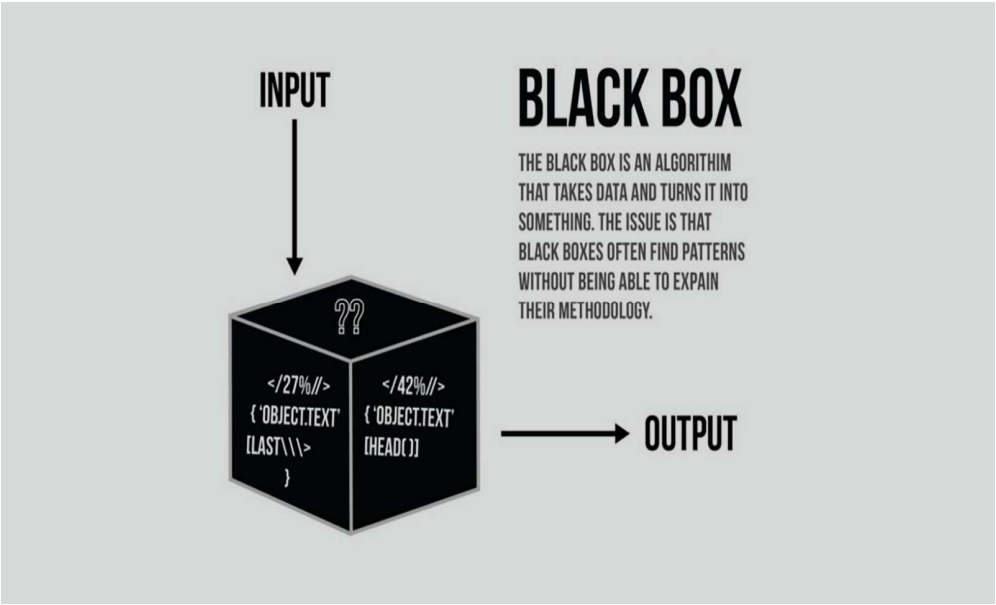
- Branch Coverage Testing >= Statement Coverage Testing

Condition Coverage

- This is closely related to decision coverage but has better sensitivity to the control flow.
- However, full condition coverage does not guarantee full decision coverage.
- Condition coverage reports the true or false outcome of each condition.
- Condition coverage measures the conditions independently of each other.

(Q.17) What is black box testing? What are the different black box testing techniques?

- Black-box testing: Testing, either functional or non-functional, without reference to the internal structure of the component or system.
- Specification-based testing technique is also known as 'black-box' or input/output driven testing techniques because they view the software as a black-box with inputs and outputs.
- The testers have no knowledge of how the system or component is structured inside the box. In black-box testing the tester is concentrating on what the software does, not how it does it.



Advantages

- Well suited and efficient for large code segments.
- Code Access not required.
- Clearly separates user's perspective from the developer's perspective through visibly defined roles.
- Large numbers of moderately skilled testers can test the application with no knowledge of implementation, programming language or operating systems.

Disadvantage

- Limited Coverage since only a selected number of test scenarios are actually performed.
- Inefficient testing, due to the fact that the tester only has limited knowledge about an application.
- Blind Coverage, since the tester cannot target specific code segments or error prone Areas. The test cases are difficult to design

.

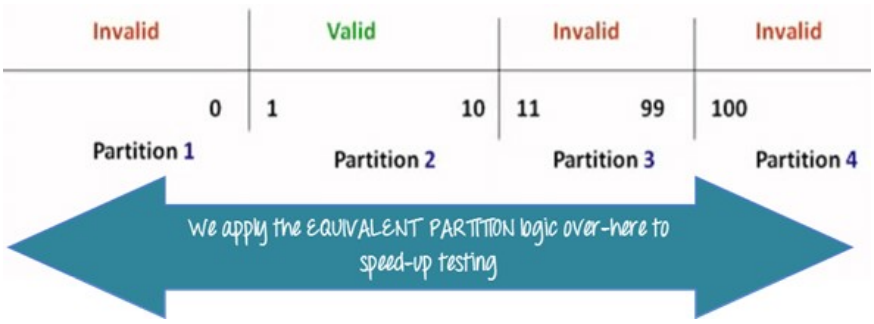
Techniques of Black Box Testing

- There are four specification-based or black-box technique:
- Equivalence partitioning
- Boundary value analysis
- Decision tables
- State transition testing

Equivalence Partitioning (E.P.)

- Aim is to treat groups of inputs as equivalent and to select one representative input to test them all
- EP can be used for all Levels of Testing
- If we want to test the following IF statement: “If value is between 1 and 100 (inclusive) (e.g value >=1 and value <=100) Then...”

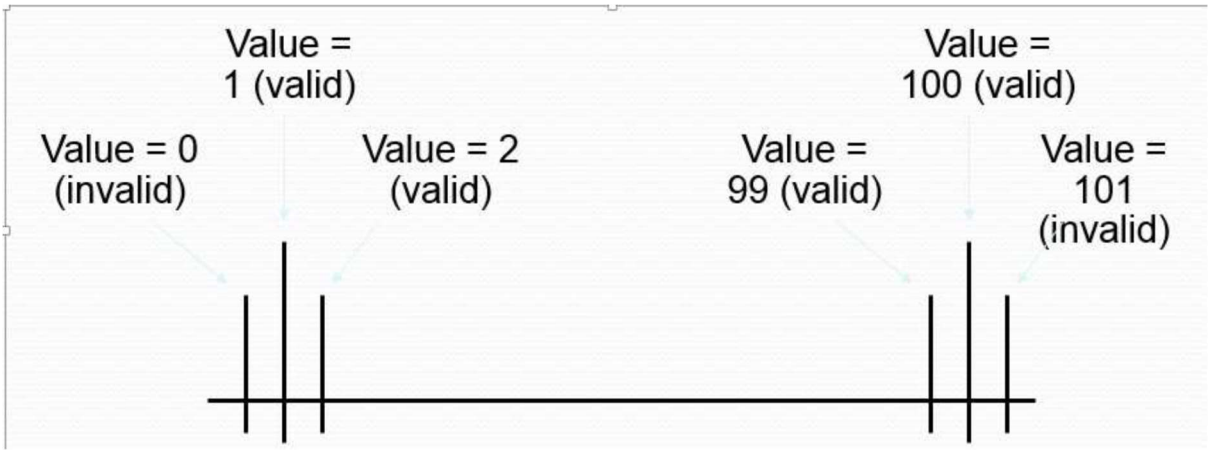
- We could put a range of numbers as shown in the below figure.



Boundary Value Analysis (B.V.A.)

- Boundary value analysis is a methodology for designing test cases that concentrates software testing effort on cases near the limits of valid ranges
- Boundary value analysis is a method which refines equivalence partitioning.

- Boundary value analysis generates test cases that highlight errors better than equivalence partitioning.
- Boundary Value Analysis (BVA) uses the same analysis of partitions as EP and is usually used in conjunction with EP in test case design



Decision Table

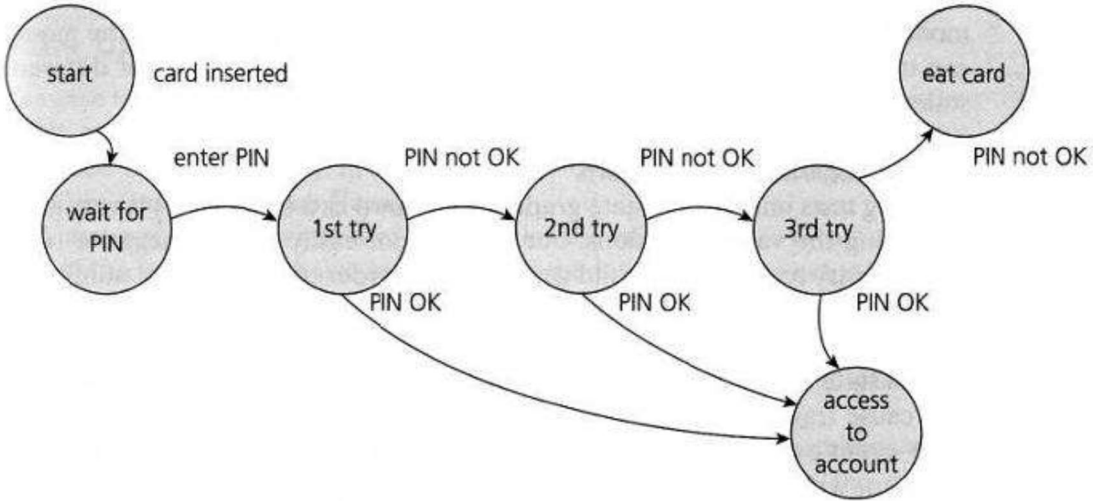
- The techniques of equivalence partitioning and boundary value analysis are often applied to specific situations or inputs.
- However, if different combinations of inputs result in different actions being taken, this can be more difficult to show using equivalence partitioning and boundary value analysis, which tend to be more focused on the user interface.
- A decision table is a good way to deal with combinations of things (e.g., inputs).
- Table based technique were
- Inputs to the system are recorded
- Outputs to the system are defined

State Transaction Testing

- State Transition Testing uses the following terms:
- **State Diagram:** A diagram that depicts the states that a component or system can assume, and shows the events or circumstances that cause and/or result from a change from one state to another.
- **State Table:** A grid showing the resulting transitions for each state combined with each possible event, showing both valid and invalid transitions.
- **State Transition:** A transition between two states of a component or system.

- **State Transition Testing:** A black box test design technique in which test cases are designed to execute valid and invalid state transitions. Also known as N-switch testing.

State Transaction Example



(Q.18) Mention what big bang testing is?

- In Big Bang integration testing all components or modules is integrated simultaneously, after which everything is tested as a whole.

Advantages:

- Convenient for small systems.

Disadvantages:

- Fault Localization is difficult.
- Since all modules are tested at once, high risk critical modules are not isolated and tested on priority. Peripheral modules which deal with user interfaces are also not isolated and tested on priority.

MODUEL :-2

(Q.19) What is the purpose of exit criteria?

How do we know when to stop testing

- Run out of time?
- Run out of budget?
- The business tells you it went live past night.
- Boss says stop.
- All defects have been criteria have been met.
- When out exit criteria have been met.
- Purpose of exit criteria is to define when we stop testing either at the end of all testing. -i.e., Product go live End of phases of testing (e.g., hand over from system test to UAT)

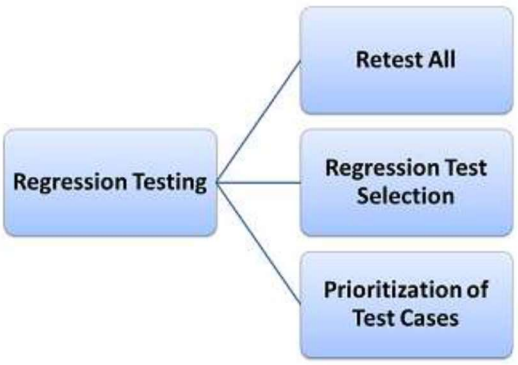
(Q.20) When should "Regression Testing" be performed?

- Regression Testing: Testing of a previously tested program following modification to ensure that defects have not been introduced or uncovered in unchanged areas of the software, as a result of the changes made. It is performed when the software or its environment is changed.

- **Regression testing should be carried out:**
- when the system is stable and the system or the environment changes
- when testing bug-fix releases as part of the maintenance phase
- It should be applied at all Test Levels

Need of Regression Testing

- Change in requirements and code is modified according to the requirement
- New feature is added to the software
- Defect fixing
- Performance issue fix



MODUEL :-2

(Q.21) What is 7 key principles? Explain in detail?

(1) Testing shaws presence of defects.

- Testing can shaws that defects are present, but cannot prove that there are no defects.
- Testing reduces the probability of undiscovered defects remaining in the software but, even if no defects are found, it is not a proof of correctness.
- We test to finds fault.
- As we find more defects the probability of undiscovered defects remaining in a system reduces.
- However testing cannot prove that there are no defects present.

(2) exhaustive testing is impossible.

- Testing everything including all combinations of inputs and pre conditions is not possible.
- So, instead of doing the exhaustive testing we can use risks and priorities to focus testing efforts.
- Exhaustive testing of complex software applications:
    - Requires enormous resources.
    - Is too expensive. Takes to long
    - It is therefore impractical.
    - Need an alternative that is pragmatic, affordable timely and provides result.
- Why do not testing everything?

Examples:

(1) system has 20 screens
(2) average 4 menus/screen
(3) average 3 options/menus
(4) average of 10 fields/screen
(5) 2 types of input for field around 100 possible values.

(3) early testing.

- Testing activities should start as early as possible in the software or system development life cycle and should be focused on defined objectives.
- Testing activities should start as early as possible in the development life cycle.

(4) defect clustering.

- A small number of modules contain most of the defects discovered during pre-release testing or are responsible for the most operational failures.
- Defects are not evenly spread in a system.

- They are clustered
- In other words, most defects found during testing are usually confined to a small number of modules.
- An important consideration in test prioritisation.

(5) pesticide paradox.

- If the same tests are repeated over and over again, eventually the same set of test cases with no longer find any new defects.
- To overcome this pesticide paradox the test cases, need to be regular reviewed and revised, and new and different tests need to we written to exercise different parts of the software or system to potentially find more defects.
- Testing identifies bugs and programmer respond to fix them.
- As bugs are eliminated by the programmers the software improves.
- As software improves the effectiveness of previous test erodes.

(6) testing is context dependent.

- Testing is basically context dependent.
- Testing is done differently in different contexts.
- Different kinds of sites are tested differently.
- **For example**
    - Safety-critical software is tested differently from an e-commerce site.
    - Whilst, testing can be 50%of development costs in NASA's apollo program it was 80% testing.
    - 3 to 10 failures per thousand lines of code typical for commercial software.
    - 1 to 3 failures per kloc typical for industrial software.
    - 0.01 failures per kloc for Nasa shuttle code.
    - Also, different industries impose different testing standards.

(7) absence of errors fallacy

- Even after defects have been resolved it may still be unusable and/or does not fulfil the users' needs and expectations.

MODUEL :-2

(Q.22) Difference between QA v/s QC v/s Tester

SR.NO.	Quality Assurance	Quality Control	Testing
1	Activities which ensure the implementation of processes, procedures and standards in context to verification of developed software and intended requirements.	Activities which ensure the verification of developed software with respect to documented (or not in some cases) requirements.	Activities which ensure the identification of bugs/error/defects in the Software
2	Focuses on processes and procedures rather than conducting actual testing on the system.	Focuses on actual testing by executing Software with intend to identify bug/defect through implementation of procedures and process.	Focuses on actual testing.
3	Process oriented activities.	Product oriented activities.	Product oriented activities.
4	Preventive activities.	It is a corrective process.	It is a preventive process.
5	It is a subset of Software Test Life Cycle (STLC).	QC can be considered as the subset of Quality Assurance.	Testing is the subset of Quality Control

MODUEL :-2

(Q.23) Difference between Smoke and Sanity?

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

(Q.24) Difference between verification and Validation.

Criteria	Verification	Validation
Definition	The process of evaluating work-products (not the actual final product) of a development phase to determine whether they meet the specified requirements for that phase.	The process of evaluating software during or at the end of the development process to determine whether it satisfies specified business requirements.
Objective	To ensure that the product is being built according to the requirements and design specifications. In other words, to ensure that work products meet their specified requirements.	To ensure that the product actually meets the user's needs, and that the specifications were correct in the first place. In other words, to demonstrate that the product fulfils its intended use when placed in its intended environment.
Question	Are we building the product, right?	Are we building the right product?
Evaluation Items	Plans, Requirement Specs, Design Specs, Code, Test Cases	The actual product/software.
Activities	• Reviews • Walkthroughs • Inspections	• Testing

MODUEL :-2

(Q.25) Explain types of Performance testing.

- Software performance testing is a means of quality assurance (QA). It involves testing software applications to ensure they will perform well under their expected workload.
- **Speed** – Determines whether the application responds quickly
- **Scalability** – Determines maximum user load the software application can handle.
- **Stability** – Determines if the application is stable under varying loads

Types of Performance Testing

- Load testing
- Stress testing
- Volume testing
- Scalability testing

(Q.26) What is Error, Defect, Bug and failure?

A mistake in coding is called errors, error found by tester is called defect, defect accepted by development team then it is called bug, build does not meet the requirements then it is failure.

(Q.27) Explain the difference between Functional testing and Non-functional testing.

SR.NO.	Functional Testing	Non-Functional Testing
1	Functional testing is performed using the functional specification provided by the client and verifies the system against the functional requirements.	Non-Functional testing checks the Performance, reliability, scalability and other non-functional aspects of the software system.
2	Functional testing is executed first	Non-functional testing should be performed after functional testing.
3	Manual testing or automation tools can be used for functional testing	Using tools will be effective for this testing.
4	Business requirements are the inputs to functional testing	Performance parameters like speed, scalability are inputs to non-functional testing.

5	Functional testing describes what the product Does	Non-functional testing describes how good the product works
6	Easy to do manual testing	Tough to do manual testing
7	Types of Functional testing are • Unit Testing • Smoke Testing • Sanity Testing • Integration Testing • White box testing • Black Box testing • User Acceptance testing • Regression Testing	Types of Non-functional testing are • Performance Testing • Load Testing • Volume Testing • Stress Testing • Security Testing • Installation Testing • Penetration Testing • Compatibility Testing • Migration Testing

(Q.28) Explain what Test Plan is? What is the information that should be covered.

- A document describing the scope approach, resources and schedule of intended test activities.
- Determining the scope and risks and identifying the objectives of testing.
- Making decisions about what to test, what roles will perform the test activities, how the test activities should be done, how the test result will be evaluated.
- scheduling test analysis and design activities.
- scheduling test implementation execution and evaluation.
- Factors which affect test planning.
- The organisations test policy.
- scope of the testing being performed.
- test objectives.
- project risk- e.g., business, technical people.
- constraints- e.g., business impulse, financial, contractual etc.
- testability availability of resource.

Test plans are continuously refined.

- As more information becomes available.
- As new risks or other are mitigated.
- Not set in concrete, but changes must be carefully managed.

Test planning activities

- **Approach:** defining approach of testing (the test strategy), including the definition of the test levels and entry and exit criteria.

Making decisions about

- what to test.
- Who do testing? (e.g., what roles will perform the test activities)
- When and how the test activities should be done and when they should be stopped. (Exit criteria see next slides)
- How the test result should be evaluated.

Test wave:

Defining the amount, level of details, structure and templates. For the test documentation.

(Q.29) What is the difference between the STLC (Software Testing Life Cycle) and SDLC (Software Development Life Cycle)?

Parameter	SDLC	STLC
Origin	Development Life Cycle	Testing Life Cycle
Definition	SDLC produces a superior-quality system that exceeds or meets users' expectations, works excellently and competently in the current, planned, and strategic information technology infrastructure, and is lucrative to manage.	STLC, on the flip side, identifies what test actions to perform and when to achieve those test activities. Although tests distinct between Organizations, there is a specific test life cycle.
Focus	On both dev (development) as well as the test process.	On merely testing process.
Performed	The stages of the SDLC are ended before those of the Software Testing Life Cycle (STLC).	The stages of the Software Testing Life Cycle (STLC) are performed after the stages of SDLC.
Objective	All through the SDLC procedure, the aim is to overcome any obstacle on the way to effective/ successful software development.	On the other hand, a test is just intended to detect pitfalls or weaknesses in the system.
Relationship with Other Life Cycle	SDLC is taken as the predecessor	STLC is taken as the successor
Team Involved	Project Managers, Business analysts, Designers, and Developers, are involved in SDLC.	Quality assurance and Testers teams are involved in Software Testing Life Cycle (STLC).
Prime goal	The prime goal is to deliver a reliable and completely functional software product.	The prime goal is to check and confirm the software meets the particular requirements and functions appropriately.
Distinct phases	It comprises stages such as requirements gathering, design,	It comprises stages such as test planning, test design, test

	execution, testing, delivery, and maintenance.	implementation, defect reporting & tracking, and test closure.
Coverage	It covers the complete software development process, from start to delivery/ deployment.	It covers the complete test process, starting from test planning to test closure.
Core Relationship	SDLC is followed by STLC to validate and verify the software product.	STLC is an integral part of SDLC, ensuring the software is thoroughly tested before deployment.
Outcome	The ultimate outcome of SDLC is delivering a higher-quality product to the customer.	The ultimate outcome of STLC is preferably to deliver bug/ flaws-free software.

(Q.30) What is the difference between test scenarios, test cases and test script?

Test scenario	Test cases	Test script
Is any functionality that can be tested.	Is a set of actions executed to verify particular features or functionality.	Is a set of instruction to test an app automatically.
Is derived from test artifacts like business requirements specifications and software requirements specifications.	Is mostly derived from test scenario.	Is mostly derived from test cases.
Help test the end-to-end functionality in an agile way.	Helps in exhaustive testing of an app.	Helps to test specific things Repeatedly.
Is more focus on what to test.	Is focused on what to test and how to test.	Is focused on expected result.
Take less time and fewer resources to create.	Require more resource and time.	Require less time for testing but more resource for scripts creating and updating.
Includes an end-to-end functionality to be tested.	Includes test steps, data, expected result for testing	Include different command to develop a script.
The main task is to check the fully functionality of a software application.	The main task is to verify compliance with the applicable standards, guidelines and customer requirements.	The main task is to verify that nothing to skipped, and the results are true as the desired test planning.
Allows quickly assessing the test scope.	Allows detecting errors and defects.	Allow carrying out an automatic execution of test case.

MODUEL :-2

(Q.31) What are the different Methodologies in Agile Development Model?

- It is a combination iterative and increment model.
- It divides the software into small incremental builds this build is provided in iterations, that means the big projects are divided into small chunks.
- Each iteration last about one to three weeks.
- Each iteration involves all the team members working simultaneously on area like planning, coding, requirement analysis, design, unit testing and acceptance testing.
- At the end of the iteration the working product is displayed to the customer or the important stake holder and it is released in the market.
- After the release we check for the feedback of the deployed software.
- If any enhancement is needed in the project, then it's done and its re-released.
- Agile model believes that every project needs to be handled differently and the existing methods need to be tailored to best suit the project requirements. In agile the tasks are divided to time boxes (small time frames) to deliver specific features for a release.
- The Agile methodology is a way to manage a project by breaking it up into several phases. It involves constant collaboration with stakeholders and continuous improvement at every stage. Once the work begins, teams' cycle through a process of planning, executing, and evaluating.

MODUEL :-2

(Q.32) when to use usability testing?

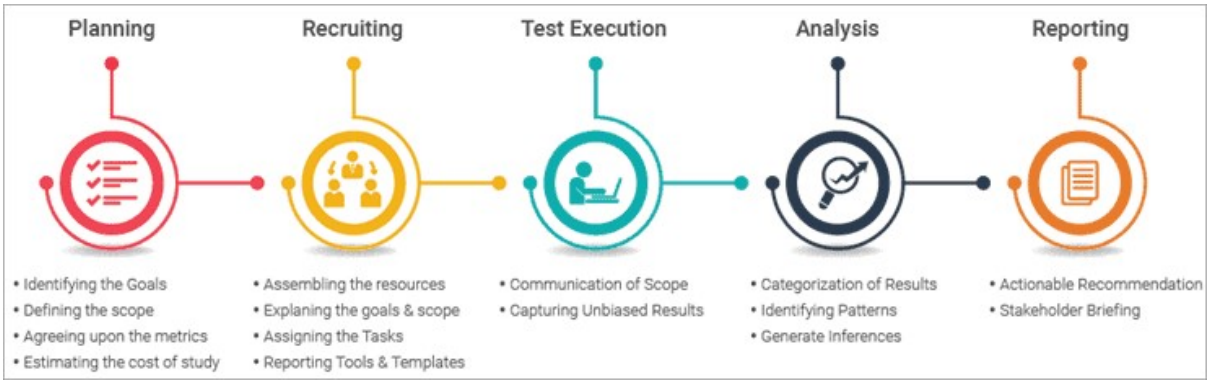
- In usability testing, you'll be looking at aspects of your web application that affect the user's experience, such as:
- How easy is it to navigate through your web application?
- Is it obvious to the user which actions are available to him or her?
- Is the look-and-feel of your web application consistent from page to page, including font sizes and colours?
- If a user forgets to fill in a required field, you might think it is a good idea to present the user with a friendly error message and change the colour of the field label to red or some other conspicuous colour.
- In Usability Testing, a small set of target end-users, of a software system, "**use**" it to expose usability defects.
- This testing mainly focuses on the user's-ease to use the application, flexibility in Handling controls and ability of the system to meet its objectives.
- This testing is recommended during the initial design phase of SDLC, which gives More visibility on the expectations of the users.

Need For Usability Testing

- Aesthetics and design are important. How well a product looks usually determines how Well, it works.
- There are many software applications / websites, which miserably fail, once launched, Due to following reasons –
- Where do I click next?
- Which page needs to be navigated?
- Which Icon or Jargon represents what?
- Error messages are not consistent or effectively displayed
- Session time not sufficient.
- Usability Testing identifies usability errors in the system early in development cycle And can save a product from failure.

- **Effectiveness of the system**
- **Efficiency**
- **Accuracy**
- **User Friendliness**
- **HOW MANY USERS DO YOU NEED?**

MODUEL :-2



(Q.33) what is the procedure for GUI testing?

- Graphical User Interface (GUI) testing is the process of testing the system's GUI of the System under Test. GUI testing involves checking the screens with the controls like menus, buttons, icons, and all types of bars – tool bar, menu bar, dialog boxes and windows etc.

- **WHAT DO YOU CHECK IN GUI TESTING?**

- Check all the GUI elements for size, position, width, length and acceptance of characters or numbers. For instance, you must be able to provide inputs to the input fields.
- Check you can execute the intended functionality of the application using the GUI
- Check Error Messages are displayed correctly
- Check for Clear demarcation of different sections on screen
- Check Font used in application is readable
- Check the alignment of the text is proper
- Check the Colour of the font and warning messages is aesthetically pleasing
- Check that the images have good clarity
- Check that the images are properly aligned
- Check the positioning of GUI elements for different screen resolution

- **MANUAL BASED TESTING**
- Under this approach, graphical screens are checked manually by testers in conformance with the requirements stated in business requirements document.
- **RECORD AND REPLAY**
- GUI testing can be done using automation tools. This is done in 2 parts. During Record, test steps are captured into the automation tool. During playback, the recorded test steps are executed on the Application under Test. Example of such tools - QTP.
- **MODEL BASED TESTING**
- A model is a graphical description of system's behaviour. It helps us to understand and predict the system behaviour. Models help in a generation of efficient test cases using the system requirements.