1) What is Exploratory Testing?

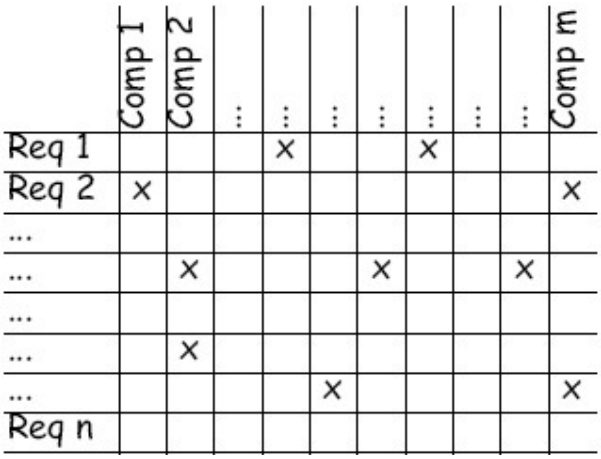
- This may be the only type of technique used for low-risk systems, but this approach may be particularly useful under extreme time pressure – in fact this is one of the factors leading to exploratory testing.

- Exploratory testing is a concurrent process where :
  - Test design, execution and logging happen simultaneously
  - Testing is often not recorded
  - Makes use of experience, heuristics 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 Adhoc testing with purpose of find bugs
  - Is structured and rigorous
  - Is cognitively (thinking) structured as compared to procedural structure of scripted testing. This structure comes from Charter, time boxing etc.
  - Is highly teachable and manageable

2) What is traceability matrix?

- To protect against changes, you should be able to trace back from every system componentto the original requirement that caused its presence.
- A software process should help you keeping the virtual table up-to-date



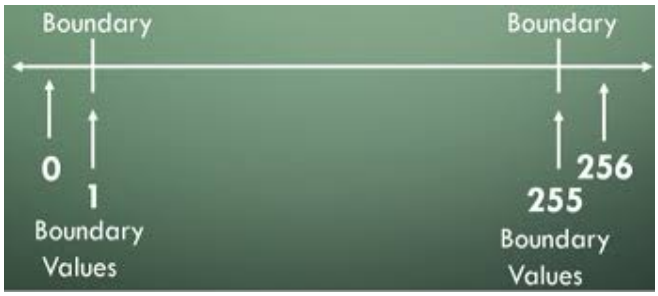
- Types of Traceability Matrix:
  1. Forward Traceability – Mapping of Requirements to Test cases
  2. Backward Traceability – Mapping of Test Cases to Requirements
  3. Bi-Directional Traceability - A Good Traceability matrix is the References from test cases to basis documentation and vice versa.

.

3) What is Boundary value testing?

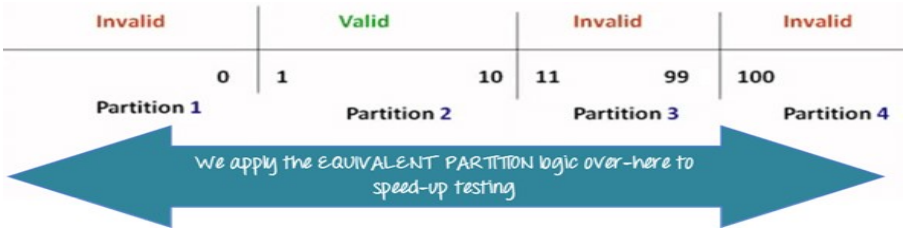
- Boundary value analysis is a methodology for designing test cases that concentrates softwaretesting effort on cases near the limits of valid ranges Boundary value analysis is a method which refines equivalence partitioning.

- Boundary value analysis generates test cases that highlight errors better than equivalence partitioning.
- The trick is to concentrate software testing efforts at the extreme ends of the equivalenceclasses.



4) What is Equivalence partitioning testing?

- 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."
- EP says that by testing just one value we have tested the partition (typically a mid-point value is used). It assumes that:
  - If one value finds 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 10.
- Plus there are 3 Invalid partitions.



5) What is Integration testing?

- Integration Testing is a level of 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** :- Testing performed to expose defects in the interfaces 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 execution of software components and proper interfacing between within the solution.

6) What determines the level of risk?

- A properly designed test that passes, reduces the overall level of Risk in a system.
- Risk – 'A factor that could result in future negative consequences; usually expressed asimpact and likelihood'.
- When testing does find defects, the Quality of the software system increases when thosedefects are fixed

- There is two types of Risks:
  1. Project Risks
  2. Product Risks

- Example of **Project risk** is Senior Team Member leaving the project abruptly.
  - Every risk is assigned a likelihood i.e. chance of it occurring, typically on a scale of 1 to 10. Also the impact of that risk is identified on a scale of 1- 10 .
  - 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

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 locationwith the involvement of developers.
- It comes under the category of both White Box Testing and Black Box Testing.

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.

9) What is component testing?

- Component(Unit) – A minimal software item that can be tested in isolation. It means "A unitis the smallest testable part of software."
- Component Testing – The testing of individual software components.
- Unit Testing is a level of the software testing process where individual units/components of asoftware/system are tested. The purpose is to validate that each unit of the software performs as designed.
- Unit testing is the 1st level of testing and is perform prior to integration testing.
- Sometimes known as "**Unit testing, module testing or program testing**".

10) What is functional system testing?

- Functional System Testing is a requirement that specifies a function that a system or systemcomponent must perform.
- A Requirement may exist as a text document and/or a model.
- Functional system testing is performed using the functional specification provided by the client and verifies the system against the functional requirement.
- Types of functional testing are :
    1. Unit testing ,
    2. Smoke testing ,
    3. Sanity testing ,
    4. Integration testing ,
    5. White box testing ,
    6. Black box testing ,
    7. User acceptance testing ,
    8. Regression testing.

11) What is Non-Functional Testing?

- Non functional testing checks performance , reliability , scalability and other non functional aspects of the software system.
- Types of the non function testing are :
    1. Performance testing ,
    2. Load testing ,
    3. Volume testing ,
    4. Stress testing ,
    5. Security testing ,
    6. Installation testing,
    7. Presentation testing ,
    8. Compatibility testing ,
    9. Migration testing.

12) What is GUI Testing?

- Graphical User Interface (GUI) testing is the process of testing the system's GUI of the

Systemunder 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,
  1. 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
  2. Check Font used in application is readable
  3. Check the alignment of the text is proper
  4. Check the Color of the font and warning messages is aesthetically pleasing
  5. Check that the images have good clarity
  6. Check you can execute the intended functionality of the application using the GUI
  7. Check Error Messages are displayed correctly

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 isvery 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 weakness.

- Types of **Adhoc Testing** are as below:

  1. **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. Buddy testing helps the testers develop better test cases and development team can also make design changes early. This testing usually happens after unit testing completion.

  2. **Pair testing :** Two testers are assigned modules, share ideas and work on the same machines to find defects. One person can execute the tests and another person can take notes on the findings. Roles of the persons can be a tester and scriber during testing.

  3. **Monkey Testing :** Randomly test the product or application without test cases with a goal to break the system.

14) What is load testing?

- Load Testing is to test the system behaviour under normal workload conditions, and it is just testing or simulating with the actual workload.
- Load testing identifies the bottlenecks in the system under various workloads and checks how the system reacts when the load is gradually increased.

- Load testing does not break the system.

15) What is stress Testing?

- Stress testing is to test the system behaviour under extreme conditions and is carried out till the system failure.
- Stress testing determines the breaking point of the system to reveal the maximum point after which it breaks.
- Stress testing tries to break the system by testing with overwhelming data or resources.

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 orsystem.
- Structure-based testing technique is also known as 'whitebox' 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.

Fig: White Box Testing

- **Test Code 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.

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

- Types of Coverage are as below:
  1. **Statement Coverage**
  2. **Decision Coverage**
  3. **Condition Coverage**

1. **Statement / Segment coverage :**
   - The statement coverage is also known as line coverage or segment coverage.
   - The statement coverage covers only the true conditions.
   - **Advantages:**
     - 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 path are tested or not.

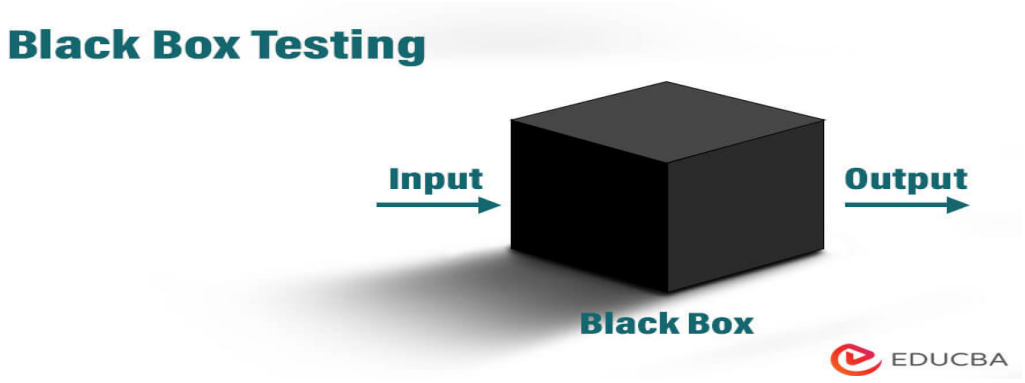
- **Disadvantages:**
  - It cannot test the false conditions.
  - It does not report that whether the loop reaches its termination condition.
  - It does not understand the logical operator.

2. **Decision Coverage / 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.
   - **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 eliminate problems that occur with statement coverage testing
   - **Disadvantages:**
     - This metric ignores branches within Boolean expressions which occur due to short circuit operators.

3. **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.

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

- Black-box testing: Testing, either functional or nonfunctional, without reference to theinternal 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.

- **Disadvantages:**
  - 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.

- There are **four** specification-based or black-box technique:
  1. **Equivalence partitioning**
  2. **Boundary value analysis**
  3. **Decision tables**
  4. **State transition testing**

1. **Equivalence partitioning :-**
   - Boundary value analysis is a methodology for designing test cases that concentrates softwaretesting 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.
   - The trick is to concentrate software testing efforts at the extreme ends of the equivalenceclasses.
2. **Boundary value analysis :-**
   - 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.
   - EP says that by testing just one value we have tested the partition (typically a mid-point value is used). It assumes that:
     - If one value finds a bug, the others probably will too
     - If one doesn't find a bug, the others probably won't either
3. **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 where :
     - Inputs to the system are recorded
     - Outputs to the system are defined
   - What will be the out come of the following Scenarios?
     - Joe is a 22 year old non smoker who goes to the gym 4 times / week and has no history of heart attacks in his family.
     - Kevin is 62 year old non smoker who swims twice a week and plays tennis. He has no history of heart attacks in his family.

	Test 1	Test 2	Test 3
> 55 yrs old	**F**	**T**	**T**
Smoker	**F**	**T**	**F**
Exercises 3 times a week +	**T**	**F**	**T**
History of Heart Attacks	**F**	**T**	**F**
Insure	**Y**	**N**	**Y**
Offer 10% Discount	**N**	**N**	**Y**
Offer 30% Discount	**Y**	**N**	**N**

4. 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**

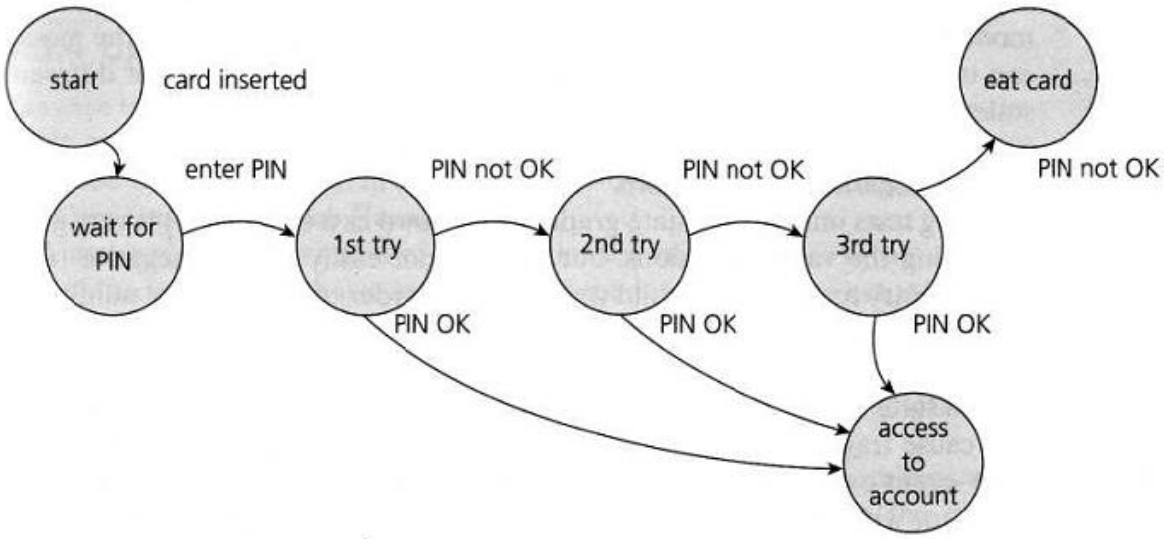


FIGURE 4.2 State diagram for PIN entry

18) Mention what bigbang testing is?

- Bing bang testing is one of the Integration Testing Method.
- In Big Bang integration testing all components or modules is integrated simultaneously, afterwhich everything is tested as a whole.

- Big Bang testing has the advantage that everything is finished before integration testingstarts.
- 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.

19) What is the purpose of exit criteria?

- Successful Testing of Integrated Application.
- Executed Test Cases are documented
- All High prioritized bugs fixed and closed
- Technical documents to be submitted followed by release Notes.
- How do we know when to stop testing?
  - Run out of time?
  - Run out of budget?
  - The business tells you it went live last night!
  - Boss says stop?
  - All defects have been fixed?
  - 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 phase of testing (e.g. hand over from System Test to UAT).

20) When should "Regression Testing" be performed?

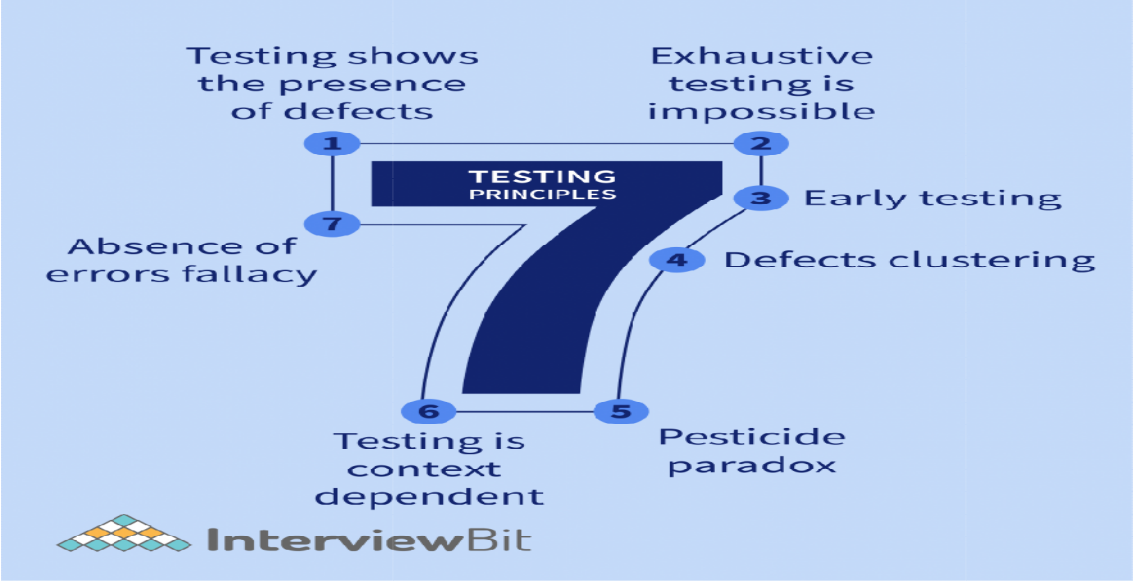
- Regression Testing: Testing of a previously tested program following modification to ensure that defects have not been introduced or uncovered in unchanged areas of the software, as result of the changes made. It is performed when the software or its environment is changed.
- Regression testing should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

21) What is 7 key principles? Explain in detail?

- General Testing Principles are as below:

1. **Testing shows presence of Defects**
2. **Exhaustive Testing is Impossible!**
3. **Early Testing**

4. **Defect Clustering**
5. **The Pesticide Paradox**
6. **Testing is Context Dependent**
7. **Absence of Errors Fallacy**



1. **Testing shows presence of Defects :**
   - Testing can show 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 find Faults.
   - 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 preconditions 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 too long
     - It is therefore impractical
     - Need an alternative that is pragmatic, affordable, timely and provides results
   - Example :
     - System has 20 screens
     - Average 4 menus / screen
     - Average 3 options / menu
     - Average of 10 fields / screen
     - 2 types of input per field
     - Around 100 possible values

- Approximate total for exhaustive testing
- 20 x 4 x 3 x 10 x 2 x 100 = 480,000 tests
- Test length = 1 sec then test duration = 17.7 days
- Test length = 10 sec then test duration = 34 weeks
- Test length = 1 min then test duration = 4 years

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

5. The Pesticide Paradox :

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

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 (KLOC) typical for commercial software
- 1 to 3 failures per KLOC typical for industrial software
- failures per KLOC for NASA Shuttle code!
- Also different industries impose different testing standards

7. Absence of Errors Fallacy :

- If the system built is unusable and does not fulfill the user's needs and expectations then finding and fixing defects does not help.
- Even after defects have been resolved it may still be unusable and/or does not fulfil the user

22) Difference between QA v/s QC v/s Tester

S/N	Quality Assurance	Quality Control	Testing
1.	Activities which ensure the implementation of processes, procedures and standards in contextto verification of developed softwareand intended requirements.	Activities which ensure theverification 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 conductingactual testing on the system.	Focuses on actual testing by executing Software withintend 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 LifeCycle (STLC).	QC can be considered asthe subset of Quality Assurance.	Testing is the subsetof Quality Control.

23) Difference between Smoke and Sanity?

S/N	Smoke Testing	Sanity Testing
1.	Smoke Testing is performed to as certainthat the critical functionalities of the program is working fine	Sanity Testing is done to check the newfunctionality / bugs have been fixed
2.	The objective of this testing is to verify "stability" of the system in order to with more rigorous testing	The objective of the testing is to verify the the"rationality" of the system in order proceed to proceed with more rigorous testing
3.	This testing is performed by the developers or testers	Sanity testing is usually performed by testers
4.	Smoke testing is usually documented or Scripted	Sanity testing is usually not documented and is unscripted
5.	Smoke testing is a subset of Regression Testing	Sanity testing is a subset of Acceptance testing
6.	Smoke testing is like General Health Check Up	Sanity Testing is like specialized health Check Up.
7.	Smoke testing exercises the entire system from end to end.	Sanity testing exercises only the particular component of the entire system

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 determinewhether they meet the specified requirements for that phase.	The process of evaluating software duringor at the end of the development processto determine whether it satisfies specifiedbusiness requirements.
Objective	To ensure that the product is being builtaccording to the requirements and design specifications. In other words, toensure that work products meet their specified requirements.	To ensure that the product actually meetsthe user's needs, and that the specifications were correct in the first place. In other words, to demonstrate thatthe product fulfills 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

25) Explain types of Performance testing.

- Software performance testing is a means of quality assurance (QA). It involves testingsoftware applications to ensure they will perform well under their expected workload.
- Types of Performance Testing
    1. **Load testing**
    2. **Stress testing**
    3. **Volume testing**
    4. **Scalability testing**

1. **Load Testing :-** It checks the product's ability to perform under anticipated user loads. The objective is to identify performance congestion before the software product is launched in the market.

2. **Stress Testing :-** It involves testing a product under extreme workloads to see whether it handles high traffic or not. The objective is to identify the breaking point of a software product.

3. **Volume Testing :-** In volume testing, large number of data is saved in a database and the overall software system's behaviour is observed. The objective is to check the product's performance under varying database volumes.

4. **Scalability Testing :-** In scalability testing, the software application's effectiveness is determined by scaling up to support an increase in user load. It helps in planning capacity additions to your software system.

26) What is Error, Defect, Bug and failure?

- A mistake in coding is called **error**.
- 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.**

27) Explain the difference between Functional testing and Non-Functional testing

S/N	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 another 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.	Nonfunctional 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 Nonfunctional testing are ▪ Performance Testing ▪ Load Testing ▪ Volume Testing ▪ Stress Testing ▪ Security Testing ▪ Installation Testing ▪ Penetration Testing ▪ Compatibility Testing ▪ Migration Testing

28) What is the difference between the STLC (Software Testing Life Cycle) and SDLC (SoftwareDevelopment Life Cycle)?

Parameter	SDLC	STLC
Origin	Development life cycle	Testing life cycle
Objective	The main project of SDLC life cycleis to complete successful development of the software including testing and other phases.	The only objective of the STLC Phaseis testing
RequirementGathering	In SDLC the business analyst gathers the requirements andcreate Development Plan	In STLC, the QA team analyze requirement documents like functional and non-functional documents and create System test Plan
High & Low Level Design	In SDLC, the development teamcreate the high and low design Plans.	In STLC, the test analyst creates theintegration test plan
Coding	The real code is developed and actual work takes place as per the design documents	The testing team prepares the testenvironment and executes them
Maintenance	SDLC Phase also includes post-deployment support and updates.	Testers, execute regression suits ,usually automation script to check maintenance code deployed.

29) What is the difference between test scenarios, test cases, and test script?

	Test scenario	Test Cases	Test Script
1.	The test scenario is just a document that is detailed and provides details about the assessment method, testing process, precondition, and anticipated output.	Test cases is a step by stepprocedure to test any functionality of the softwareapplication/ product.	Test script is set of instruction or a short program to test any functionality of softwareapplication/ product.
2.	The test scenarios are the ones based on the use situation and give one-lineinformation one what to check.	Test cases is a manual approachof software testing.	Test script is an automaticapproach of software testing.

	Test scenario	Test Cases	Test Script
4.	These are high level actions.	Point by point test case configuration encourages tester to test viably.	Automatic testing approach is beneficial for constant execution.
5.	Writing the test scenario'sprimary objective is an address end to get rid of functionality of a software program.	Test cases are written bymanually.	Test scripting is done by scripting format.
6.	It will take less time as compared to test cases.	Test case is developed in form of templates.	Test script is developed in form of scripting.
7.	The test scenario will helpus in a way that is nimble of through the functionality.	If the tester does not have a good understanding of how theprogram is used or about the recent risks to the program, then it will be difficult to usethe test cases properly.	Active software projectsfrequently change. so testers have to make a continuous effort to update the scripts to match the changes of the new product.
8.	Test scenario are reallyeasy to maintain due to their highlevel design.	Test case is used in manualtesting environment.	Test script is used inautomatic testing environment.
9.	The test scenarios tend to be work on the essential to"things to be tested".	Test cases are classified as delegated, positive, reusable,negative and UI test cases.	Test script are characterized as manual test script and automatictest scripts.

30) 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, and how the test results will be evaluated?
- Scheduling test analysis and design activities.
- Test planning activities:
- **Approach**: Defining the overall approach of testing (the test strategy), including the definition of the test levels and entry and exit criteria.
- Integrating and coordinating the testing activities into the software life cycle activities: acquisition, supply, development, operation and maintenance.
- Making decisions about:
  - **what** to test
  - **Who** do testing? i.e. what roles will perform the test activities
  - **When** and how the test activities should be done and when they should be stopped
  - **how** the test results will be evaluated
- Assigning resources for the different tasks defined.
- **Test ware**: Defining the amount, level of detail, structure and templates for the test

documentation.

- **Process**: Setting the level of detail for test procedures in order to provide enough information to support reproducible test preparation and execution.

31) When to used Usability Testing?

- Usability testing is one of the non functional testing.
- When software is ready, it is important to make sure that the user experience with the product should be seamless. It should be easy to navigate and all the functions should be working properly, the competitor's website will win the race. Therefore, usability testing is performed.
- The objective of usability testing is to understand customers' needs and requirements and also how users interact with the product (software). With the test, all the features, functions, and purposes of the software are checked.
- The primary goals of usability testing are – discovering problems (hidden issues) and opportunities, comparing benchmarks, and comparison against other websites.
- The parameters tested during usability testing are efficiency, effectiveness, and satisfaction.
- It should be performed before any new design is made. This test should be iterated unless all the necessary changes have been made.
- Improving the site consistently by performing usability testing enhances its performance which in return makes it the best website.

32) What is the procedure for GUI Testing?

- Below are approaches for GUI testing :

1. **Manual based testing :**
   - Under this approach, graphical screens are checked manually by testers in conformance with the requirements stated in business requirements document.

2. **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.

3. **Model based testing :**
   - A model is a graphical description of system's behavior. It helps us to understand and predict the system behaviour. Models help in a generation of efficient test cases using the system requirement.

33) 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.