1. **What is component testing?**
   - Component testing is a minimal software item that can be tested in isolation. It means "a unit is the smallest testable part of software"
   - Component testing is the testing of individual software component
   - Component testing is a level of the software testing process where individual unit/component of the software/system are tested. The purpose is to validate that each unit of the software perform as designed
   - Component tests are typically written and run by software developer to ensure that the code meets its design and behave as intended with debugging tool
   - Component test find problems early in the development cycle
   - Component testing is performed by using the white box testing method

2. **What is functional system testing?**
   - Functional System Testing is requirement that specifies a function that a system or system component must perform
   - A Requirement may exist as a text document and/or a model
   - There is two types of Test Approach
     1. Requirement Based Functional Testing
     2. Process based testing
   - Functional System Testing Functionality As below:
     a. **Accuracy**: Provision of right or agreed results or effects
     b. **Interoperability**: Ability to interact with specified systems
     c. **Compliance**: Adhere to applicable standards, conventions, regulations or laws
     d. **Auditability**: Ability to provide adequate and accurate audit data
     e. **Suitability**: Presence and appropriateness of functions for specified tasks

3. **What is Non-functional testing?**
   - Non-Functional Testing is 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.

4. **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.
   - Error guessing can be done by the people having enough experience on the system to "guess" the most likely source of errors.

5. **What is exploratory testing?**
    - 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
    - Is not a technique but it is an approach. What actions you perform next is governed by what you are doing currently

6. **What is traceability matrix?**
    - 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.

7. **What is boundary value testing?**
    - Boundary value testing is a methodology for designing test cases that concentrates software testing effort on cases near the limits of valid ranges.
    - Boundary value testing is a method which refines equivalence partitioning.
    - The trick is to concentrate software testing efforts at the extreme ends of the equivalence classes.

8. **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.
    - EP says that by testing just one value we have tested the partition (typically a mid-point value is used).
    - It assumes that:
        a. If one value finds a bug, the others probably will too.
        b. If one doesn't find a bug, the others probably won't either.

9. **What is white box testing and list the types of white box testing?**
    - White Box Testing is based on an analysis of the internal structure of the component or system.
    - 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.
    - Types of white box testing are:

10. **What is black box testing? What are the different black box testing techniques?**
    - Black-box testing is either functional or non-functional, without reference to the internal structure of the component or system.
    - 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.
    - There are four black box testing techniques:
        a. Equivalence partitioning
        b. Boundary value analysis
        c. Decision tables
        d. State transition testing

11. **Mention what big bang testing is?**
    - In Big Bang testing all components or modules is integrated simultaneously, after which everything is tested as a whole.
    - Big Bang testing has the advantage that everything is finished before integration testing starts.
    - The major disadvantage is that in general it is time consuming and difficult to trace the cause of failures because of this late integration.
    - Here all component are integrated together at once, and then tested.

12. **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 last night
    - Boss says stop
    - All defects have been fixed
    - When out exit criteria have been met

    - **Exit Criteria typically measures:**

    - Thoroughness measures, such as coverage of requirements or of code or risk coverage

    - Estimates of defect density or reliability measures. (e.g. how many defects open by category)

    - Cost.

    - Residual Risks, such as defects not fixed or lack of test coverage in certain areas.

    - Schedules - such as those based on time to market.

13. **What is 7 key principles? Explain in detail?**
    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.

2. **Exhaustive Testing is Impossible:**

- Testing everything including all combinations of inputs and preconditions is not possible.
- Exhaustive testing of complex software applications:
  - ➔ requires enormous resources
  - ➔ is too expensive
  - ➔ takes too long
- It is therefore impractical

3. **Early Testing:**

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

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
- Similarly, most operational failures of a system are usually confined to a small number of modules
- An important consideration in test prioritisation.

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 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 (KLOC) 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.

14. Difference between QA v/s QC v/s Tester

Sr. No	Quality Assurance	Quality Control	Tester
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.	Process oriented activities.	Product oriented activities.
4	Preventive activities.	It is a corrective process.	It is a corrective 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.

15. Difference between Smoke and Sanity?

Smoke	**Sanity**
Smoke Testing is performed to check critical functionalities	Sanity Testing is done to check the new functionality / bugs have been fixed
This testing is performed by the developer or tester	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 part of Regression testing	Sanity testing is a part 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 advance health check up

16. **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 fulfill 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

17. **Explain types of Performance testing.**

1. **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.
- Load testing is a kind of performance testing which determines a system's performance under real-life load conditions. This testing helps determine how the application behaves when multiple users access it simultaneously.

2. **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.
- Stress testing is used to test the stability & reliability of the system. This test mainly determines the system on its robustness and error handling under extremely heavy load conditions.

3. **Endurance Testing**
4. **Spike Testing**
5. **Volume Testing**
6. **Scalability Testing**

18. **What is Error, Defect, Bug and failure?**

Error: A discrepancy between a computed, observed, or measured value or condition and the true, specified, or theoretically correct value or condition. This can be a misunderstanding of the internal state of the software, an oversight in terms of memory management, confusion about the proper way to calculate a value, etc.

Defect: Commonly refers to several troubles with the software products, with its external behaviour or with its internal features.

Bug: A fault in a program which causes the program to perform in an unintended or unanticipated manner. See: anomaly, defect, error, exception, and fault. Bug is terminology of Tester.

Failure: The inability of a system or component to perform its required functions within specified performance requirements. See: bug, crash, exception, and fault.

19. **Difference between Priority and Severity**
    - **High Priority & High Severity:** An error which occurs on the basic functionality of the application and will not allow the user to use the system. (Eg. A site maintaining the student details, on saving record if it, doesn't allow to save the record then this is high priority and high severity bug.)
    - **High Priority & Low Severity:** The spelling mistakes that happens on the cover page or heading or title of an application.
    - **High Severity & Low Priority:** An error which occurs on the functionality of the application (for which there is no workaround) and will not allow the user to use the system but on click of link which is rarely used by the end user.
    - **Low Priority and Low Severity:** Any cosmetic or spelling issues which is within a paragraph or in the report (Not on cover page, heading, title).

20. **What is Bug Life Cycle?**
    - A computer bug is an error, flaw, mistake, failure, or fault in a computer program that prevents it from working correctly or produces an incorrect result. Bugs arise from mistakes and errors, made by people, in either a program's source code or its design.
    - The duration or time span between the first time defects is found and the time that it is closed successfully, rejected, postponed or deferred is called as Bug Life Cycle.

21. **What is Integration testing?**
    - Testing performed to expose defects in the interfaces and in the interactions between integrated components or systems
    - Integration Testing is a level of the software testing process where individual units are combined and tested as a group.
    - The purpose of this level of testing is to expose faults in the interaction between integrated units. Test drivers and test stubs are used to assist in Integration Testing.

22. **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 always performed in Virtual Environment.
    - It is always performed within the organization.
    - It is the form of Acceptance Testing.

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

23. **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 performed in Real Time Environment.
    - It is always performed outside the organization.
    - It is also the form of Acceptance Testing.
    - It is only a kind of Black Box Testing.

24. **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.

25. **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 breaking the system under various workloads and checks how the system reacts when the load is gradually increased
    Load testing does not break the system

26. **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 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.

27. **When should "Regression Testing" be performed?**
    - 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
    - It should be considered complete when agreed completion criteria for regression testing have been met
    - Regression test suites evolve over time and given that they are run frequently are ideal candidates for automation

28. **Explain the difference between Functional testing and Non-Functional testing**

Functional Testing	Non-Functional Testing
Functional testing is performed using the functional specification provided by the client and verifies the system against the functional requirements.	Non-Functional testing check the Performance, reliability, scalability and other non-functional aspects of the software system.
Functional testing is executed first	Non functional testing should be performed after functional testing

Manual testing or automation tools can be used for functional testing	Using tools will be effective for this testing
Business requirements are the inputs to functional testing	Performance parameters like speed , scalability are inputs to non-functional testing.
Functional testing describes what the product does	Nonfunctional testing describes how good the product works
Easy to do manual testing	Tough to do manual testing
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

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

Test Scenario:

A Scenario is any functionality that can be tested.

It is also called Test Condition, or Test Possibility.

Test Scenario is 'What to be tested'

Test scenario is nothing but test procedure.

The scenarios are derived from use cases.

Test Scenario represents a series of actions that are associated together.

Test Case

Test cases involve the set of steps, conditions and inputs which can be used while performing the testing tasks.

Test Case is 'How to be tested'

Test case consist of set of input values, execution precondition, expected Results and executed post-condition developed to cover certain test Condition.

Test cases are derived (or written) from test scenario.

Test Case represents a single (low level) action by the user.

Test cases are set of input and output given to the System.

Test Script

A test script in software testing is a set of instructions that will be performed on the system under test to test that the system functions as expected.

There are various means for executing test scripts.

- Manual Testing
- Automation Testing

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.

- Defining the overall approach of testing (the test strategy), including the definition of the test levels and entry and exit criteria.
- 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.
- Scheduling test implementation, execution and evaluation.

Factors which affect test planning

- The organisation's test policy
- Scope of the testing being performed
- Testing objectives
- Project Risks – e.g. business, technical, people
- Constraints – e.g. business imposed, financial, contractual etc
- Criticality (e.g. system/component level)
- Testability
- Availability of resources

31. What is priority?

Priority is Relative and Business-Focused. Priority defines the order in which we should resolve a defect. Should we fix it now, or can it wait? This priority status is set by the tester to the developer mentioning the time frame to fix the defect. If high priority is mentioned then the developer has to fix it at the earliest. The priority status is set based on the customer requirements.

Priority can be of following types:

- **Low:** The defect is an irritant which should be repaired, but repair can be deferred until after more serious defect has been fixed.
- **Medium:** The defect should be resolved in the normal course of development activities. It can wait until a new build or version is created.
- **High:** The defect must be resolved as soon as possible because the defect is affecting the application or the product severely. The system cannot be used until the repair has been done.
- **Critical:** Extremely urgent, resolve immediately

32. What is severity?

Severity is absolute and Customer-Focused. It is the extent to which the defect can affect the software. In other words it defines the impact that a given defect has on the system.

Severity can be of following types:

- **Critical:** The defect that results in the termination of the complete system or one or more component of the system and causes extensive corruption of the data. The failed function is unusable and there is no acceptable alternative method to achieve the required results then the severity will be stated as critical.
- **Major (High):** The defect that results in the termination of the complete system or one or more component of the system and causes extensive corruption of the data. The failed function is unusable but there exists an acceptable alternative method to achieve the required results then the severity will be stated as major.
- **Moderate (Medium):** The defect that does not result in the termination, but causes the system to produce incorrect, incomplete or inconsistent results then the severity will be stated as moderate.

- **Minor (Low):** The defect that does not result in the termination and does not damage the usability of the system and the desired results can be easily obtained by working around the defects then the severity is stated as minor.
- **Cosmetic:** The defect that is related to the enhancement of the system where the changes are related to the look and field of the application then the severity is stated as cosmetic.

33. **Advantage of Bugzilla**
    Bugzilla is a defect tracking tool, however it can be used as a test management tool as such it can be easily linked with other test case management tools like Quality Centre, Test link etc.
    This open bug-tracker enables users to stay connected with their clients or employees, to communicate about problems effectively throughout the data management chain.
    Key features of Bugzilla includes
    Advanced search capabilities
    E-mail Notifications
    Modify/file Bugs by e-mail
    Time tracking
    Strong security
    Customization
    Localization