Q. 1: Why a software needs to be tested?

Every software product needs to be tested since, the development 'process' is unable to produce defect free software. Even if the development process is able to produce a defect free software, we will not be able to know unless & until we test it. Without testing it, we shall not be having enough confidence that it will work.

Testing not only identifies and reports defect but also measures the quality of the product, which helps to decide whether to release the product, or not.

<<<<<< ========== >>>>>>

Q. 2: What is the reason that Software have Bugs?

Following factors contribute to the presence of bugs in the software applications.

1) Software development tools like visual tools, class libraries, compilers, scripting tools, etc. usually introduce their own bugs in the system.

2) To err is human. Likewise programmers do make mistakes while programming.

3) In fast-changing business environments continuously modified requirements are becoming a fact of life. Such frequent changes requested by the customer leads to errors in the application already nearing completion. Last minute design changes leads to many chaos like redesign of the whole system, rescheduling of engineers, scrapping of the work already completed, fresh requirements of compatible hardware etc.

4) A quickly written but poorly documented code is bound to have bugs. It becomes difficult to maintain and modify such code that is badly written or poorly documented.

- it's tough to maintain and modify code that is badly written or poorly documented; the result is bugs. In many organizations management provides no incentive for programmers to document their code or write clear, understandable, maintainable code. In fact, it's usually the opposite: they get points mostly for quickly turning out code, and there's job security if nobody else can understand it ('if it was hard to write, it should be hard to read').

5) When project deadlines come too close & time pressures come, mistakes are bound to come.

<<<<<< ========== >>>>>>

Q. 3: What is the difference between QA and Testing?

QA stands for "Quality Assurance", and focuses on "Prevention" of defects in the product being developed. It is associated with the "Process" and activities related to the Process Improvement. Quality Assurance measures the quality of the processes employed to create a quality product.

Whereas "Testing" refers to "Quality Control", and focuses on Detection of Defect and removal thereafter. Or Quality Control measures the quality of a product.

<<<<<< ==================== >>>>>>

Q. 4: What is the difference between Software Testing and Debugging?

Testing is the process of locating or identifying the errors or bugs in a software system.

Whereas Debugging is the process of Fixing the identified Bugs. It involves a process of analyzing and rectifying the syntax errors, logic errors and all other types of errors identified during the process of testing.

<<<<<< ==================== >>>>>>

Q. 5: What is the difference between a Bug and a Defect?

"Bug" is a problem or an error in the software code, which is found in the application during Testing. Bug is responsible for failure of the application to comply with the desired specifications.

Whereas "Defect" is problem reported by the customer during usage of the software application.

<<<<<< ==================== >>>>>>

Q. 6: What is the difference between a Bug and an Enhancement?

"Bug" is a problem or an error in the software code, which is found in the application during Testing. Bug is responsible for failure of the application to comply with the desired specifications.

Whereas "Enhancement" is the additional feature or functionality found and added to the application as desired by the end user / real word customer or tester during the testing process.

<<<<<< ==================== >>>>>>

Q. 7: What is the difference between Requirements & Specifications?

"Requirements" are statements given by the customer as to what needs to be achieved by the software system. Later on these requirements are converted into specifications which are nothing but feasible or implementable requirements.

Whereas "Specifications" are feasible requirements derived from various statements given by the customer. These are the starting point for the product development team.

<<<<<< ==================== >>>>>>

Q. 8: What is the sequence of succession in STLC - Software Testing Life Cycle?

1) Test Planning

2) Test Analysis

3) Test Design

4) Construction and verification

5) Testing Cycles

6) Final Testing and Implementation and Post Implementation.

<<<<<< ==================== >>>>>>

Q. 9: What is the difference between Verification and Validation?

"Verification" involves reviews and meetings to evaluate documents, plans, code, requirements, and specifications to confirm whether items, processes, services, or documents conform to specified requirements or not. This can be done with the help of checklists, issues lists, walkthroughs, and inspection meetings. The purpose of verification is to determine whether the products of a given phase of the software development cycle fulfill the requirements established during the previous phase or not.

Whereas "Validation" is the determination of the correctness of the final program or software product produced from a development project with respect to the user needs and requirements. This involves actual testing of the product and takes place after verifications are completed.

"Software Verification" raises the question, "Are we building the Product Right?"; that is, does the software conform to its specification.

"Software Validation" raises the question, "Are we building the Right Product?"; that is, is the software doing what the user really requires.

<<<<<< ==================== >>>>>>

Q. 10: What is the difference between a Test Plan and a Use Case?

"Test Plan" is a document describing an introduction to the client company, intended scope, overview of the application, test strategy, schedule of testing activities, roles and responsibilities, deliverables and milestones. It describes test items, features to be tested, testing tasks, details of the personnel performing each task and any risks requiring contingency planning.

Whereas a "Use Case" describes the process as to how an end user uses a specific functionality in the application. It is a summary of user actions and system response to the user actions. It contains the flows like typical flow, alternate flow and exceptional flow. It also contains pre condition and post condition.

<<<<<< ==================== >>>>>>

Q. 11: What is the difference between Bug Priority & Bug Severity?

"Bug Priority" is the need on how urgently bug is needed to be fixed. It describes the importance of the bug. Bug priority may change according to the schedule of testing.

Whereas "Bug Severity" is the quantum of danger as to how badly the bug can harm the system. It describes as to how bad the bug is. Severity is a feature of constant nature associated with the bug.

<<<<<< ========== >>>>>>

Q. 12: What is difference between Waterfall Model and V Model?

"Waterfall Model" Is a sequential software development model (a process for the creation of software) in which development is seen as flowing steadily downwards (like a waterfall) through the phases of requirements analysis, design, implementation, testing (validation), integration, and maintenance. To follow the waterfall model, we proceed from one phase to the next in a purely sequential manner. In traditional waterfall model, testing comes at the fag end of the development process.

Whereas "V Model" or "Life Cycle Testing" involves carrying out verification of consistency, completeness and correctness of software at every stage of the development life cycle. It aims at catching the defects as early as possible and thus reduces the cost of fixing them. It involves continuously testing the system during all stages of the development process rather than just limiting testing to the last stage.

<<<<<< ========== >>>>>>

Q. 13: What are Baseline Documents?

Baseline documents are the documents, which have been approved by the customer and will not have any more changes. Baseline Documents cover all the details of the project and have undergone "walkthrough" process. Once a document is Base-lined it cannot be changed unless there is a change request duly approved by the customer. Service Level Agreement (SLA) & Business Requirement Documents (BRD) are the examples of Baseline Documents.

<<<<<< ========== >>>>>>

Q. 14: What is Defect Density?

"Defect Density" Is a software metric defined as: Total number of defects per LOC (lines of code). Alternatively It can be: Total number of defects per Size of the Project. Here the measure of "Size of the Project" can be number of Function Points, Number of Feature Points, number of Use Cases or KLOC (Kilo Lines of Code) etc

<<<<<< ========== >>>>>>

Q. 15: What is Negative Testing?

"Negative Testing" involves testing the application for failure like conditions. It involves testing the tool with improper inputs. For example entering the special characters in place of a phone number.

<<<<<< ==================== >>>>>>

Q. 16: What is Incremental Integration Testing?

"Incremental Integration Testing" Involves continuous testing of an application while new functionality is simultaneously added. It requires that various aspects of an application's functionality be independent enough to work separately before all parts of the program are completed. This testing is done either by programmers or by testers.

<<<<<< ==================== >>>>>>

Q. 17: What is the difference between Unit Testing, Component Testing and Integration Testing?

"Unit Testing" involves testing of individual programs, modules, or components to demonstrate that the program executes as per the specification and it validates the design and technical quality of the application. In Unit Testing, the Called Components (or Communicating Components) are replaced with Stubs, Simulators, or Trusted Components. Testing Stubs or Drivers are used to simulate the behavior of interfacing modules.

"Component Testing" is like "Unit Testing" with the difference that all Stubs and Simulators are replaced with the real objects. Here a Unit is a component, and integration of one or more such components is also a Component.

Whereas "Integration Testing" is the test process which begins after two or more programs components have been successfully unit tested. It is conducted by the development team to validate the interaction or communication/flow of information between the individual components that will be integrated.

<<<<<< ==================== >>>>>>

Q. 18: What is the difference between Statement Coverage, Branch Coverage and Path Coverage?

"Statement Coverage" is a type of "White-Box Testing" technique, involving execution of all statements at least once. Statement coverage is a simple metric to calculate & measure the number of statements in a method or class which have been executed. Its key benefit is its ability to identify which blocks of code have not been executed.

"Branch Coverage" is an outcome of a decision, and measures the number of decision outcomes or branches, which have been tested. This takes a more in-depth view of the source code rather than a simple "Statement Coverage". A branch is an outcome of a decision. For example Boolean decisions like an "If - Statement", has two outcomes or branches (i.e. True and False).

Whereas "Path Coverage" is a method of testing which satisfies the coverage criteria

through which the program is tested across each logical path. Usually, paths through the

program are grouped into a finite set of classes and one path out of every class is tested. In Path Coverage flow of execution takes place from the start of a method to its exit. Path Coverage ensures that we test all decision outcomes independently of one another.

<<<<<< ========== >>>>>>

Q. 19: What is the difference between Ad-hoc Testing, Monkey Testing and Exploratory Testing?

"Ad-hoc Testing" is performed without any planning of process and without any documentation like Test Case or Test Scenarios. It involves test design and simultaneous test execution. For Ad-hoc testing the testers possess significant understanding of the software before testing it.

"Monkey Testing" is done with no specific test in mind. Here the monkey is the producer of any input data (which can be either a file data or can be an input device data). It involves pressing some keys randomly and checking whether the software fails or not.

Whereas "Exploratory Testing" involves simultaneous learning, test design and test execution. It is a type of "Ad-hoc Testing", but only difference is that in this case, the tester does not have much idea about the application & he explores the system in an attempt to learn the application and simultaneously test it.

<<<<<< ========== >>>>>>

Q. 20: What is the difference between System Testing and End-to-End Testing or E2E Testing?

"System Testing" falls within the scope of Black-Box testing and the tester requires no knowledge of the inner design of the code or logic. It is conducted on a complete / combined part of a system to verify that all-functional, information, structural and quality requirements as per the specifications have been met.

"End-to-End Testing" or "E2E Testing" is also quite similar to "System Testing". It involves testing of the application in a environment that simulates the real-world use, such as interacting with a database, using network communications, or interacting with other hardware, applications, or systems.

<<<<<< ========== >>>>>>

Q. 21: What is the difference between Alpha Testing and Beta Testing?

Typically a software product passes through two stages of testing before it is considered to be Final. The first stage is known as "Alpha Testing". It is often performed by potential users / customers or an independent test team at the developers' site. It is usually done when the development of the software product is nearing completion; minor design changes may still be made as a result of Alpha testing.

The second stage coming after alpha testing is known as "Beta Testing". Versions of the software, known as beta versions, are released to a limited audience outside of the

programming team so that further evaluation by the users can reveal more faults or bugs in

the product. Sometimes, beta versions are made available to the open public to increase the feedback field to a maximum number of future users.

<<<<<< ========== >>>>>>

Q. 22: What is the difference between Static Testing and Dynamic Testing?

"Static Testing" involves testing activities performed without actually running the software. It includes Document review, code inspections, walkthroughs and desk checks etc.

Whereas "Dynamic Testing" Is used to describe the testing of the dynamic behavior of the software code. It involves actual compilation & running of the software by giving input values and checking if the output is as expected. It is the validation portion of Verification and Validation.

<<<<<< ========== >>>>>>

Q. 23: What is the difference between Smoke Testing and Sanity Testing?

The general term of "Smoke Testing" has come from leakage testing of sewers & drain lines involving blowing smoke into various parts of the sewer and drain lines to detect sources of unwanted leaks and sources of sewer odors. In relation to software testing field, Smoke testing Is a non-exhaustive software testing, ascertaining that the most crucial functions of the program work well, without getting bothered about finer details of it.

Whereas "Sanity Testing" Is an initial testing effort to find out if the new software version is performing well enough to accept it for a major testing effort. For example, if the new software is crashing the systems every 5 minutes, bogging down the systems to a crawl, or destroying the databases, then it can be concluded that the software may not be in a 'sane' enough condition to warrant further testing in its current state.

<<<<<< ========== >>>>>>

Q. 24: What is the difference between Stress Testing and Load Testing?

"Stress Testing" is subjecting a system to an unreasonable load while denying it the adequate resources as required to process that load. The resources can be RAM, disc space, mips & interrupts etc. etc. The idea is to stress a system to the breaking point in order to find bugs, which will make the break potentially harmful. The system is not expected to process the overload without adequate resources, but to fail in a decent manner (e.g., failure without corrupting or losing data). In stress testing the load (incoming transaction stream) is often deliberately distorted so as to force the system into resource depletion.

Whereas "Load Testing" is a test performed with an objective to determine the maximum sustainable load which the system can handle. Load is varied from a minimum (zero) to the maximum level the system can sustain without running out of resources or causing excessive delay in transactions.

<<<<<< ========== >>>>>>

Q. 25: What is the difference between Black Box Testing & White Box Testing?

First of all Black-Box and White-Box both are Test Design Methods.

"Black-Box" test design treats the system as a "Black-Box" (Wherein the tester can't see as to what is there inside the box). Hence we design the test cases in such a way that we pour the input from one end of the box and expect a certain specific output from the other end of the box. To run these test cases, the tester need not know as to how the input gets transformed to output inside the box. Black-Box is also known as Behavioral-Box or Functional-Box or Opaque-Box or Gray-Box or Closed-Box.

Whereas "White-Box" test design treats the system as a Transparent Box, which allows anyone to see inside the "Box". In White-Box the tester is able to see the process of transformation of an "Input" into an "Output" inside the box. Hence we design the test cases with a view to test the internal Logic, Paths or Branches of the box. White-Box is also known as Structural-Box or Glass-Box or Clear-Box or Translucent-Box test design

<<<<<< ==================== >>>>>>

Q. 26: What is Quality?

Quality software is software that is reasonably bug-free, delivered on time and within budget, meets requirements and expectations and is maintainable. However, quality is a subjective term. Quality depends on who the customer is and their overall influence in the scheme of things.

Customers of a software development project include end-users, customer acceptance test engineers, testers, customer contract officers, customer management, the development organization's management, test engineers, testers, salespeople, software engineers, stockholders and accountants. Each type of customer will have his or her own slant on quality. The accounting department might define quality in terms of profits, while an end-user might define quality as user friendly and bug free.

<<<<<< ==================== >>>>>>

Q. 27: What is an Inspection?

An inspection is a formal meeting, more formalized than a walkthrough and typically consists of 3-10 people including a moderator, reader (the author of whatever is being reviewed) and a recorder (to make notes in the document). The subject of the inspection is typically a document, such as a requirements document or a test plan. The purpose of an inspection is to find problems and see what is missing, not to fix anything. The result of the meeting is documented in a written report. Attendees should prepare for this type of meeting by reading through the document, before the meeting starts; most problems are found during this preparation. Preparation for inspections is difficult, but is one of the most cost-effective methods of ensuring quality, since bug prevention is more cost effective than bug detection.

<<<<<< ==================== >>>>>>

Q. 28: What is Good Design?

Design could mean to many things, but often refers to functional design or internal design. Good functional design is indicated by software functionality can be traced back to customer and end-user requirements. Good internal design is indicated by software code whose overall structure is clear, understandable, easily modifiable and maintainable; is robust with sufficient error handling and status logging capability; and works correctly when implemented.

<<<<<< ==================== >>>>>>

Q. 29: What is Six Sigma?

"Six Sigma" means Six Standard Deviations from the mean. It is a methodology aimed to reduce defect levels below 3.4 Defects Per one Million Opportunities. Six Sigma approach improves the process performance, decreases variation and maintains consistent quality of the process output. This leads to defect reduction and improvement in profits, product quality and customer satisfaction.

<<<<<< ==================== >>>>>>

Q. 30: What is difference between CMM and CMMI?

"CMM" means "Capability Maturity Model" developed by the Software Engineering Institute (SEI). It is a process capability maturity model, which aids in the definition and understanding of an organization's processes. CMM is intended as a tool for objectively assessing the ability of government contractors' processes to perform a contracted software project.

Whereas "CMMI" means "Capability Maturity Model Integration" & it has superceded CMM. The old CMM has been renamed to Software Engineering CMM (SE-CMM).

<<<<<< ==================== >>>>>>

Q. 31: What is Verification?

Verification ensures the product is designed to deliver all functionality to the customer; it typically involves reviews and meetings to evaluate documents, plans, code, requirements and specifications; this can be done with checklists, issues lists, walkthroughs and inspection meetings.

<<<<<< ==================== >>>>>>

Q. 32: What is Validation?

Validation ensures that functionality, as defined in requirements, is the intended behavior of the product; validation typically involves actual testing and takes place after verifications are completed.

<<<<<< ==================== >>>>>>

Q. 33: What is a Test Plan?

A software project test plan is a document that describes the objectives, scope, approach and focus of a software testing effort. The process of preparing a test plan is a useful way to think through the efforts needed to validate the acceptability of a software product. The completed document will help people outside the test group nderstand the why and how of product validation. It should be thorough enough to be useful, but not so thorough that none outside the test group will be able to read it.

<<<<<< ========== >>>>>>

Q. 34: What is a Walkthrough?

A walkthrough is an informal meeting for evaluation or informational purposes. A walkthrough is also a process at an abstract level. It's the process of inspecting software code by following paths through the code (as determined by input conditions and choices made along the way). The purpose of code walkthroughs is to ensure the code fits the purpose. Walkthroughs also offer opportunities to assess an individual's or team's competency.

<<<<<< ========== >>>>>>

Q. 35: What is Software Life Cycle?

Software life cycle begins when a software product is first conceived and ends when it is no longer in use. It includes phases like initial concept, requirements analysis, functional design, internal design, documentation planning, test planning, coding, document preparation, integration, testing, maintenance, updates, re-testing and phase-out.

<<<<<< ========== >>>>>>

Q. 36: What is the Difference between STLC & SDLC?

STLC means " Software Testing Life Cycle". It starts with activities like : 1) Preparation of Requirements Document 2) Preparation of Test Plan 3) Preparation of Test Cases 4) Execution of Test Cases 5) Analysis of Bugs 6) Reporting of Bugs 7) Tracking of Bugs till closure.

Whereas SDLC means " Software Development Life Cycle" is a software development process, used by a systems analyst to develop an information system. It starts with activities like :

1) Project Initiation

2) Requirement Gathering and Documenting

3) Designing

4) Coding and Unit Testing

5) Integration Testing

6) System Testing

7) Installation and Acceptance Testing

8) Support or Maintenance

<<<<<< ==================== >>>>>>

Q. 37: What are the various components of STLC?

Various components of "Software Testing Life Cycle" are

1) Requirements Document

2) Preparation of Test Plan

3) Preparation of Test Cases

4) Execution of Test Cases

5) Analysis of Bugs

6) Reporting of Bugs

7) Tracking of Bugs till closure

<<<<<< ==================== >>>>>>

Q. 38: What is the Difference between Project and Product Testing?

If any organization is developing the application according to the client specification then it is called as project. Accordingly its testing is known as "Project Testing"

Whereas If any organization is developing the application and marketing it is called as product. Hence its testing is known as "Product Testing"

<<<<<< ==================== >>>>>>

Q. 39: What are the Testing Types & Techniques?

Black Box and White Box are the most popular types of software testing. These are not the stand-alone testing techniques.

Testing techniques falling under the Black-Box type are: 1) Equivalence Partitioning 2) Boundary Value Analysis 3) Cause-Effect Graphing 4) Error-Guessing etc.

Whereas testing techniques falling under the White-Box type are:

1) Statement coverage

2) Decision coverage

3) Condition coverage

4) Decision-condition coverage

5) Multiple condition coverage

6) Basis Path Testing

7) Loop testing

8) Data flow testing etc.

<<<<<< ==================== >>>>>>

Q. 40: How do you introduce a new software QA process?

It depends on the size of the organization and the risks involved. For large organizations with high-risk projects, a serious management buy-in is required and a formalized QA process is necessary. For medium size organizations with lower risk projects, management and organizational buy-in and a slower, step-by-step process is required.

Generally speaking, QA processes should be balanced with productivity, in order to keep any bureaucracy from getting out of hand. For smaller groups or projects, an ad-hoc process is more appropriate. A lot depends on team leads and managers, feedback to developers and good communication is essential among customers, managers, developers, test engineers and testers. Regardless the size of the company, the greatest value for effort is in managing requirement processes, where the goal is requirements that are clear, complete and testable.

<<<<<< ==================== >>>>>>

Q. 41: What are the common problems coming across Software Development Process?

Common problems in software development process are:

1) Poor Projection of Requirements - Generally the users are not very clear in regards to their exact needs. Most of the specifications given to Software Development Outsourcing vendors are rough and very sketchy. Problems arise if the requirements are unclear, incomplete, too general, and not testable etc.

2) Miscommunication - Becomes the main cause of problem when the developers remain ignorant of the exact needs or expectations of the customer.

3) Unrealistic Schedules - Cause problems if too much work is crammed in too little time.

4) Inadequate Testing - Problems arise when the application has not been adequately tested before giving it to the customer & the customer complains after using it or when there is a systems crash.

<<<<<< ==================== >>>>>>

Q. 42: What is the role of documentation in QA?

Documentation plays a critical role in QA. QA practices should be documented, so that they are repeatable. Specifications, designs, business rules, inspection reports, configurations, code changes, test plans, test cases, bug reports, user manuals should all be documented. Ideally, there should be a system for easily finding and obtaining

of documents and determining what document will have a particular piece of information.

<<<<<< ==================== >>>>>>

Q. 43: What is Phase Containment and Defect Prevention?

Phase Containment is incorporating QA into all the phases of SDLC. It results in Defect Prevention. If QA team performs Requirements Review, Design Review and Code Review, defects would be few when actual application is tested. That means we have prevented many defects by performing reviews at each stage of SDLC.

<<<<<< ==================== >>>>>>

Q. 44: Why a Developer should not Test?

Of course, a developer can test, but he can't be a good tester. If developers do the testing of their own work or of the work of their peers, then the following problems crop up

1) Misunderstandings of the requirements or specifications may go unnoticed.

2) Under a given time frame, usual tendency of developers is to allocate more time in improving the code or documentation rather than doing the testing of the code.

3) Developers have a tendency of being optimistic of producing a defect free code hence 'under' test the product.

4) Testing needs great skill, while an occasional tester with no prior training in testing techniques is no match to a trained bug hunter whose sole activity is testing.

5) To uncover large number of bugs, tester needs to be aggressive. Whereas developer will not be aggressive, if he is testing his own product.

Testers are rewarded if they hunt lots of bugs, developers are rewarded if the product they developed has less number of bugs and this balance can only be maintained if the separate teams exist for testing and development.

<<<<<< ==================== >>>>>>

Q. 45: Out of Tester & Developer, who is most appropriate to do Unit Testing & Integration Testing?

Of course Developers.

It is quite difficult for a tester to do unit testing and integration testing since it involves in-depth understanding of the code. Hence developers should do the unit testing and integration testing. While doing unit testing & integration testing, a few misinterpretation of requirements might escape from the notice of the developer, but its better to test with these issues rather than not testing at all.

For a better success, code developed by one developer, then his peer should do the unit test.

<<<<<< ========== >>>>>>

Q. 46: What are the important Check Points for Installation Testing?

Installation testing of a new software application should be able to check points like

1) To check previously installed versions of the software or its dependent software or patches like previously installed Service Packs. This is to ensure that older version of the software should not get installed over the newer version.

2) Installer should be able to define Default Installation Path like "C:\Program Files\."

3) Installer should be able to allow the user to install the new software at a location different from the Default Installation Path.

4) To check if the product can be installed "Over the Network"

5) Installation should start automatically when the CD is inserted in the CD drive.

6) Software Remove / Repair options should be available to the user while using the Installer.

7) While uninstalling the software, check that all the registry keys, files, Dll, shortcuts, active X components are removed from the system.

8) To check if the product can be installed without Administrative Privileges (login as guest).

9) To check if the product can be installed on different operating systems.

10) To check if the product can be installed on a system having non-compliant configuration like with less Memory / RAM / Hard Disc Capacity.

<<<<<< ========== >>>>>>

Q. 47: What is Installation Testing?

"Installation Testing" is performed to ensure that all the Installed features and options of the software are functioning properly. Its main objective is to verify that all necessary components of the application are actually installed or not without missing out any component.

<<<<<< ========== >>>>>>

Q. 48: Do automated testing tools make testing easier?

For larger projects, or ongoing long-term projects, they can be valuable. But for small projects, the time needed to learn and implement them is usually not worthwhile. A common type of automated tool is the record/playback type. For example, a test engineer clicks through all combinations of menu choices, dialog box choices, buttons, etc. in a GUI and has an automated testing tool record and log the results. The recording is typically in the form of text, based on a scripting language that the testing tool can interpret. If a change is made (e.g. new buttons are added, or some underlying code in the application is changed), the application is then retested by just playing back the recorded actions and compared to the logged results in order to check effects of the change.

<<<<<< ========== >>>>>>

Q. 49: What should be done after a bug is found?

When a bug is found, it needs to be communicated and assigned to developers that can fix it.

After the problem is resolved, fixes should be re-tested. Additionally, determinations should be made regarding requirements, software, hardware, safety impact, etc., for regression testing to check the fixes didn't create other problems elsewhere.

If a problem-tracking system is in place, it should encapsulate these determinations. A variety of commercial, problem-tracking/management software tools are available. These tools, with the detailed input of software test engineers, will give the team complete information so developers can understand the bug, get an idea

of its severity, reproduce it and fix it.

<<<<<< ========== >>>>>>

Q. 50: What to do when software is full of bugs & it can't be tested at all?

In this situation the best solution is to have test engineers go through the process of reporting whatever bugs or problems initially show up, with the focus being on critical bugs.

Since this type of problem can severely affect schedules and indicates deeper problems in the software development process, such as insufficient unit testing, Insufficient integration testing, poor design, improper build or release procedures, managers should be notified and provided with some documentation as evidence of the problem.

<<<<<< ========== >>>>>>

Q. 51: What is Parallel Testing?

Parallel testing involves testing multiple products or sub-components simultaneously. A parallel test station typically shares a set of test equipment across multiple test sockets, but, in some cases, it may have a separate set of hardware for each unit under test (UUT).

The majority of nonparallel test systems test only one product or sub-component at a time, leaving expensive test hardware idle more than 50 percent of the test time. Thus, with parallel testing, you can increase the throughput of manufacturing test systems without spending a lot of money to duplicate and fan out additional test systems.

<<<<<< ========== >>>>>>

Q. 52: What is Comparison Testing?

Comparison testing is testing that compares software weaknesses and strengths to those of competitors' products.

<<<<<< ========== >>>>>>

Q. 53: What is Probe Testing?

It is almost same as Exploratory testing. It is a creative, intuitive process. Everything testers do is optimized to find bugs fast, so plans often change as testers learn more about the product and its weaknesses.

Session-based test management is one method to organize and direct exploratory testing. It allows us to provide meaningful reports to management while preserving the creativity that makes exploratory testing work. This page includes an explanation of the method as well as sample session reports, and a tool we developed that produces metrics from those reports.

<<<<<< ========== >>>>>>

Q. 54: What questions you would ask to yourself while deciding to automate the tests?

Best approach would be to raise the following questions:

1) Automating this test and running it once will cost more than simply running it manually once. How much more?

2) An automated test has a finite lifetime, during which it must recoup that additional cost. Is this test likely to die sooner or later? What events are likely to end it?

3) During its lifetime, how likely is this test to find additional bugs (beyond whatever bugs it found the first time it ran)? How does this uncertain benefit balance against the cost of automation?

<<<<<< ========== >>>>>>

Q. 55: What do we lose with Automation compared to Manual Testing?

Creating an automated test is usually more time-consuming & costly than running it once manually. The cost differential varies, depending on the product and the automation style.

1) If the product is being tested through a GUI and your automation style is to write scripts that drive the GUI, an automated test may be several times as expensive as a manual test.

2) If you use a GUI capture / replay tool that tracks your interactions with the product and builds a script from them, automation is relatively cheaper. It is not as cheap as manual testing, though, when you consider the cost of recapturing a test from the beginning after you make a mistake, the time spent organizing and documenting all the files that make up the test suite, the aggravation of finding and working around bugs in the tool, and so forth. Those small "in the noise" costs can add up surprisingly quickly.

3) If you're testing a compiler, automation might be only a little more expensive than manual testing, because most of the effort will go into writing test programs for the compiler to compile. Those programs have to be written whether or not they're saved for reuse.

<<<<<< ========== >>>>>>

Q. 56: What is the difference between Structural testing & functional testing?

Structural testing examines how the program works, taking into account possible pitfalls in the structure and logic.

Functional testing examines what the program accomplishes, without regard to how it works internally.

<<<<<< ========== >>>>>>

Q. 57: What is the difference between code coverange analysis & test coverage analysis?

Both these terms are similar. Code coverage analysis is sometimes called test coverage analysis. The academic world generally uses the term "test coverage" whereas the practitioners use the term "code coverage".

<<<<<< ========== >>>>>>

Q. 58: What are the basic assumptions behind coverage analysis?

Following assumptions tell us about the strengths and limitations of coverage analysis technique.

1) Bugs relate to control flow and you can expose Bugs by varying the control flow. For example, a programmer wrote "if (c)" rather than "if (!c)".

2) You can look for failures without knowing what failures might occur and all tests are reliable, in that successful test runs imply program correctness. The tester understands what a correct version of the program would do and can identify differences from the correct behavior.

3) Other assumptions are achievable specifications, no errors of omission, and no unreachable code.

<<<<<< ==================== >>>>>>

Q. 59: What are main advantages of statement coverage metric of software testing?

1) The main advantage of statement coverage metric is that it can be applied directly to object code and does not require processing source code. Usually the performance profilers use this metric.

2) Bugs are evenly distributed through code; therefore the percentage of executable statements covered reflects the percentage of faults discovered.

<<<<<< ==================== >>>>>>

Q. 60: What are the drawbacks of statement coverage metric of software testing?

1) It is insensitive to some of the control structures.

2) It does not report whether loops reach their termination condition - only whether the loop body was executed. With C, C++, and Java, this limitation affects loops that contain break statements.

3) It is completely insensitive to the logical operators (|| and &&).

4) It cannot distinguish consecutive switch labels.

<<<<<< ==================== >>>>>>

Q. 61: What are advantages & drawbacks of decision coverage metric of software testing?

Decision coverage has the main advantage of simplicity & is free from many problems of statement coverage.

Disadvantage of decision coverage is that this metric ignores branches within boolean expressions which occur due to short-circuit operators.

<<<<<< ==================== >>>>>>

Q. 62: What is multiple condition coverage metric of software testing?

Multiple condition coverage reports whether every possible combination of boolean sub-expressions occurs. 100% multiple condition coverage implies 100% condition determination coverage.

Drawback of this metric is that it becomes tedious to find out the minimum number of test cases required, especially for very complex boolean expressions. Another drawback of this metric is that the number of test cases required can vary to a large extent among various conditions having similar complexity.

<<<<<< ========== >>>>>>

Q. 63: What are advantages & drawbacks of path coverage metric of software testing?

Advantages are:

1) Path coverage requires extremely thorough testing.

Disadvantages are:

1) Since loops introduce an unbounded number of paths, this metric considers only a limited number of looping possibilities.

2) The number of paths is exponential to the number of branches. For example, a function containing 10 if-statements has 1024 paths to test. Adding just one more if-statement doubles the count to 2048.

3) Many paths are impossible to exercise due to relationships of data.

<<<<<< ========== >>>>>>

Q. 64: What is the best sequence of coverage goals as implementation strategy

1) Invoke at least one function in 90% of the source files (or classes).

2) Invoke 90% of the functions.

3) Attain 90% condition/decision coverage in each function.

4) Attain 100% condition/decision coverage.

<<<<<< ========== >>>>>>

Q. 65: What is configuration Management?

Configuration Management is a discipline applying technical and administrative direction and surveillance to: identify and document the functional and physical characteristics of a configuration item, control changes to those characteristics, record and report change processing and implementation status, and verify compliance with specified requirements.

<<<<<< ========== >>>>>>

Q. 66: What is "Key Word Driven" or "Test Plan Driven" Method of Testing?

This method uses the actual Test Case document developed by the tester using a spreadsheet containing special "Key-Words".

In this method, the entire process is data-driven, including functionality. The Key Words control the processing.

<<<<<< ========== >>>>>>

Q. 67: Is the tool going to replace the testers any day?

This is not even remotely true. The automated testing tool is just another tool that will allow testers to do their jobs better by:

1) Performing the boring-type test cases that they now have to do over and over again

2) Freeing up some of their time so that they can create better, more effective test cases

The testers are still going to perform tests manually for specific application changes. Some of these tests may be automated afterward for regression testing.

<<<<<< ========== >>>>>>

Q. 68: What is Automated Testing?

"Automated Testing" is automating the manual testing process currently in use. This requires that a formalized "manual testing process" currently exists in your company. "Automated Testing" process includes:

1) Detailed test cases, including predictable "expected results", which have been developed from Business Functional Specifications and Design documentation.

2) A standalone Test Environment, including a Test Database that is restorable to a known constant, such that the test cases are able to be repeated each time there are modifications made to the application.

<<<<<< ========== >>>>>>

Q. 69: What is the purpose of Automated Test Tools?

The real use and purpose of automated test tools is to automate regression testing.

This means that we must have or must develop a database of detailed test cases that are repeatable, and this suite of tests is run every time there is a change to the application to ensure that the change does not produce unintended consequences.

< < < < < < = > > > > > >

Q. 70: What is a Traceability Matrix?

Traceability means that we would like to be able to trace back and forth how and where any work product fulfills the directions of the preceding product.

The matrix deals with the where, while the how we have to do ourselves, once we know the where.

<<<<<< ==================== >>>>>>

Q. 71: What is a Data Flow Diagram (DFD)?

Data Flow Diagram is a graphical representation of the "flow" of data through an information system. A data flow diagram can also be used for the visualization of data processing. It is common practice for a designer to draw a context-level DFD first which shows the interaction between the system and outside entities.

<<<<<< ==================== >>>>>>

Q. 72: What is the Difference between Retest and Regression Testing?

When a bug is fixed by the developer, testing the same bug to ensure whether it has been fixed or not is known as retesting.

Whereas testing the other features of the application which might be affected by the bug fix is known as regression testing.

<<<<<< ==================== >>>>>>

Q. 73: What is a Test Scenario?

Test Scenario is a set of test cases which ensure that the business process flows are tested from end to end. They may be independent tests or a series of tests that follow each other, each dependent on the output of the previous one.

The terms "test scenario" and "test case" are often used synonymously.

<<<<<< ==================== >>>>>>

Q. 74: What is Statement Coverage In software testing,?

Statement coverage is one of the ways of measuring code coverage. It describes the degree to which the software code of a program has been tested.

All the statements in the code must be executed and tested.

<<<<<< ==================== >>>>>>

Q. 75: What is What is Defect Leakage?

Defect leakage refers to the defect Found \ reproduced by the Client or User, which the tester was unable to found.

<<<<<< ==================== >>>>>>

Q. 76: What is the difference between Functional Testing and System Testing?

Functionality testing is based on functional requirements of the application whereas the system testing is end to end testing it covers all the functionality, performance, usability, database, stress testing.

Functional testing is the subset of system testing, but both are Black box testing.

<<<<<< ==================== >>>>>>

Q. 77: What Is a Test Bed?

Test bed is the environment which is required to test the software.

This includes requirement of Hardware, Software, Memory, CPU speed, Operating system etc.

<<<<<< ==================== >>>>>>

Q. 78: What is Latent Bug?

Latent Bug is a bug, which gets unobserved in two or more releases of the application.

<<<<<< ==================== >>>>>>

Q. 79: What is Bi-directional Traceability Matrix?

Bidirectional Traceability refers to the Forward and Backward traceability.

Forward Traceability is from requirements to design to code to testcases.

Whereas the Backward Traceability is in the reverse direction, meaning thereby the end product has met the requirements or not. It is quite difficult to achieve the Backward traceability without the help of a tool.

Bidirectional Traceability is the prime expectation of CMMI.

<<<<<< ==================== >>>>>>

Q. 80: What is Base Lining?

Baselining is the process by which the quality and cost effectiveness of a service is assessed, usually in advance of a change to the service.

Baselining usually includes comparison of the service before and after the Change or analysis of trend information.

<<<<<< ========== >>>>>>

Q. 81: What is configuration Management?

Configuration Management (or CM) is the processes of controlling, coordinating and tracking the Standards and procedures for managing changes in an evolving software product.

Configuration Testing is the process of checking the operation of the software being tested on various types of hardware.

<<<<<< ========== >>>>>>

Q. 82: What is the role of QA in a software producing company?

QA is responsible for managing, implementing, maintaining and continuously improving the Processes in the Company and enable internal projects towards process maturity and facilitate process improvements and innovations in the organization.

Tester is responsible for carrying out the testing efforts in the company.

In many companies QA person is responsible both the roles of Testing as well as creating and improving the processes.

<<<<<< ========== >>>>>>

Q. 83: What is Fuzz Testing?

Fuzz testing a technique of testing an application by feeding random inputs.

<<<<<< ========== >>>>>>

Q. 84: What is Failure Mode and Effect Analysis (FMEA)?

Failure Mode and Effect Analysis is a systematic approach to risk identification and analysis of identifying possible modes of failure and attempting to prevent their occurrence.

<<<<<< ========== >>>>>>

Q. 85: What is Path Testing?

Path Testing or Path Coverage is a white box method of testing which satisfies the coverage criteria through which the program is tested across each logical path. Usually, paths through the program are grouped into a finite set of classes and one path out of every class is tested.

In Path Coverage flow of execution takes place from the start of a method to its exit. Path Coverage ensures that we test all decision outcomes independently of one another

<<<<<< ==================== >>>>>>

Q. 86: What is Test Maturity Model or TMM?

Test Maturity Model or TMM is a five level staged framework for test process improvement, related to the Capability Maturity Model (CMM) that describes the key elements of an effective test process.

<<<<<< ==================== >>>>>>

Q. 87: What is Back-To-Back Testing?

Back-To-Back Testing refers to the testing process in which two or more variants of a component or system are executed with the same inputs, the outputs compared, and analyzed in cases of discrepancies.

<<<<<< ==================== >>>>>>

Q. 88: What is a Blocked Test Case?

Blocked Test Case refers to the test case, which cannot be executed because the preconditions for its execution are not fulfilled.

<<<<<< ==================== >>>>>>

Q. 89: What is the difference between API & ABI?

Application Programming Interface (API) is a formalized set of software calls and routines that can be referenced by an application program in order to access supporting system or network services.

Whereas Application Binary Interface (ABI) is a specification defining requirements for portability of applications in binary forms across different system platforms and environments.

<<<<<< ==================== >>>>>>

Q. 90: What is I V & V?

I V & V means Independent Verification and Validation.

Verification typically involves reviews and meetings to evaluate documents, plans, code, requirements, and specifications. Verification can be done with the help of checklists, issues lists, walkthroughs, and inspection meetings.

Whereas Validation typically involves actual testing and takes place after verifications are completed.

<<<<<< ==================== >>>>>>

Q. 91: What is LCSAJ?

LCSAJ means "Linear Code Sequence And Jump". It consists of the following three items like:

1) The start of the linear sequence of executable statements.

2) The end of the linear sequence.

3) The target line to which control flow is transferred at the end of the linear sequence.

<<<<<< ==================== >>>>>>

Q. 92: What is Measure of Completeness in software testing?

In software testing there are two measures of completeness, code coverage and path coverage.

Code coverage is a white box testing technique to determine how much of a program's source code has been tested. There are several fronts on which code coverage is measured. Code coverage provides a final layer of testing because it searches for the errors that were missed by the other test cases.

Whereas Path coverage establishes whether every potential route through a segment of code has been executed and tested.

<<<<<< ==================== >>>>>>

Q. 93: What is Pair Programming?

Pair Programming is a software development approach whereby lines of code of a component are written by two programmers sitting at a single computer. This means ongoing real-time code reviews are performed.

<<<<<< ==================== >>>>>>

Q. 94: What is N+1 Testing?

N+1 Testing is a variation of Regression Testing. It involves testing conducted with multiple cycles in which errors found in test cycle 'N' are resolved and the solution is re-tested in test cycle N+1. The cycles are typically repeated until the solution reaches a steady state and there are no errors.

<<<<<< ==================== >>>>>>

Q. 95: What is Error Seeding?

Error Seeding is the process of intentionally adding known defects to those already in the component or system for the purpose of monitoring the rate of detection and removal, and estimating the number of remaining defects.

<<<<<< ========== >>>>>>

Q. 96: What is Extreme Programming?

Extreme Programming called XP in short is a deliberate and disciplined approach to software development.

XP is extremely successful since it lays maximum stress on the customer satisfaction. XP methodology is designed to deliver the software as per the customer needs and at a time when it is needed.

While following XP methodology the developers confidently respond to fast changing requirements of the customer may be quite late in the life cycle.

XP methodology lays emphasis on an excellent team work. Managers, customers, and developers are all part of a team dedicated to delivering quality software.

<<<<<< ========== >>>>>>

Q. 97: What is the difference between an application server and a Web server?

Web server serves pages for viewing in a Web browser, while an application server provides methods that client applications can call.

A little more precisely, we can say that, a Web server exclusively handles HTTP requests, whereas an application server serves business logic to application programs through any number of protocols.

<<<<<< ========== >>>>>>

Q. 98: What is Cause Effect Analysis?

Cause effect analysis is an approach for studying the specifications carefully and identifying the combinations of input conditions or causes and their effect in the form of a table and designing test cases.

It is suitable for applications in which combinations of input conditions are few and readily visible.

<<<<<< ========== >>>>>>

Q. 99: What is Error Guessing?

Error guessing is a supplementary technique of test case design involving test case design based on the tester's intuition and experience. There is no formal procedure. However, a checklist of common errors is taken for reference.

<<<<<< ==================== >>>>>>

Q. 100: What is Basis Path Testing?

Basis Path Testing is white box testing method involving design of test cases to cover every statement, every branch and every condition in the code which has been written.

This method attempts statement coverage, decision coverage and condition coverage

<<<<<< ==================== >>>>>>

Q. 101: What is the difference a Software Tester & Testing Analyst?

Testing analysts are more commonly involved with tasks at a higher level of abstraction, such as test process design, test planning, and test case design.

Whereas Software Testers may be involved with test case design and test procedure construction, and interaction with the actual software systems.

<<<<<< ==================== >>>>>>

Q. 102: What are Software Testing Specialities?

Testing specialties include test automation, load testing, usability testing, testing methodology, software inspections, industry or application expertise, test metrics, test management, white box testing & security testing etc.

<<<<<< ==================== >>>>>>

Q. 103: What can be the various Job Levels in the Software Testing Domain in a Company?

Various job levels within the testing domain can include the tester, test analyst, test manager or test specialist, test consultant or Test executive.

<<<<<< ==================== >>>>>>

Q. 104: What is a Test Suite?

Set of collection of test cases is called a test suite.

It contains more detailed instructions or goals for each collection of test cases. It contains a section where the tester identifies the system configuration used during testing. It may also contain prerequisite states or steps, and descriptions of the tests as well.

< < < < < < = > > > > > >

Q. 105: What is a scenario test?

This is a test based on a hypothetical story used to help a person think through a complex problem or system.

Generally scenario test have following five key characteristics.

1) A story

2) Which is motivating

3) Which is credible

4) Which is complex

5) Which is easy to evaluate.

Scenario tests are different from test cases in a way that test cases cover single steps whereas scenarios cover a number of steps. Test suites and scenarios can be used together for a complete system test.

<<<<<< ==================== >>>>>>

Q. 106: What is Defect Tracking?

In engineering practice, defect tracking is the process of finding defects in a product by the process of inspection, testing, or recording feedback from customers, and tracking them till their closure.

In software engineering, defect tracking is of significant importance, since complex software systems have thousands of defects due to which their management, evaluation and prioritizing is a difficult task. Hence defect tracking systems in software engineering are computer database systems which store defects and help people to manage them.

<<<<<< ==================== >>>>>>

Q. 107: What is Formal Verification in context with Software & Hardware systems?

Formal verification is the process of proving or disproving the correctness of a system with respect to a certain formal specification or property, with the help of formal methods. Generally the formal verification is carried out algorithmically.

Approaches to implement formal verification are :

1) State space enumeration

2) Symbolic state space enumeration

3) Abstract interpretation

4) Abstraction refinement

5) Process-algebraic methods

6) Reasoning with the help of automatic theorem provers like HOL or Isabelle.

<<<<<< ==================== >>>>>>

Q. 108: What is the concept of Fuzz Testing?

Fuzz testing is a software testing technique involving attachment of the inputs of a program to a source of random data. Main advantage of fuzz testing is that the test design is extremely simple, and remains free of preconceptions about system behavior.

Fuzz testing is generally used in large software development projects which use black box testing. Fuzz testing provides a high benefit to cost ratio.

Fuzz testing technique is also used for the measurement of quality of large software systems. The advantage is that the cost of generating tests is relatively low.

Fuzz testing is helps to enhance the software security and software safety because it often finds odd oversights and defects which normal human testers would fail to find, and even the most careful human test designers would fail to create tests for.

Fuzz testing is not a substitute for exhaustive testing or formal methods; it can only provide a random sample of the system's behavior. Passing a fuzz test may only indicate that a particular software is capable to handles exceptions without crashing and it may not indicate its correct behavior.

<<<<<< ==================== >>>>>>

Q. 109: What are the different forms of fuzz testing?

1) Valid fuzz Testing to assure that the random input is reasonable, or conforms to actual production data.

2) Simple fuzz Testing usually uses a pseudo random number generator to provide an input.

3) A combined approach uses valid test data with some proportion of totally random input injected.

By using all the above techniques in combination, fuzz-generated randomness can test the un-designed behavior surrounding a wider range of designed system states.

<<<<<< ==================== >>>>>>

Q. 110: What is a Web Application & How does it look like?

A web application is an internet based application, consisting of a set many scripts, which are normally stored on some Web server and are made to interact with some databases or any other similar sources of the dynamic content.

Web applications provide an interactive Form to the user, wherein feeds inputs according to the fields provided in the form; then he clicks on a button like "Submit" or "OK" to store his inputs on the database & perform a set of calculations & present back the desired information.

Web Applications are becoming popular since these are a via media for exchange of information between various service providers and respective customers across the internet. These web applications are by & large not dependent on any platform. Popular examples of Web applications are Google / Yahoo or similar search engines, Internet Banking websites of several Banks, E-mail facility providing sites like Gmail, Yahoo Mail, Rediff Mail etc., Sale & Purchase sites like E-Bay etc.

<<<<<< ========== >>>>>>

Q. 111: What is Server Side Includes or SSI?

Server Side Includes or SSI is a mechanism by which we can include files using a special type of HTML comment which is similar to the include feature of todays scripting languages like JSP & PHP etc.

Old type CGI programs and ASP scripts are still using Server Side Includes or SSI to include libraries of code.

<<<<<< ========== >>>>>>

Q. 112: What is the difference between Dynamic Analysis & Static Analysis?

Dynamic Analysis: refers to the process of testing and evaluation of a program by executing data in real-time. The objective is to find errors in a program while it is running, rather than by repeatedly examining the code offline. Smoke testing is a type of dynamic analysis.

Static Analysis: refers to a set of techniques of program analysis where the program is not actually executed rather it is analyzed by some tools to produce the desired information. Objective of performing static analysis to ensure soundness and completeness of the program.

<<<<<< ========== >>>>>>

Q. 113: What is Vulnerability Analysis?

Vulnerability Analysis is a process which defines, identifies, and classifies the security holes or vulnerabilities in a computer, network, or communications infrastructure. It can be used to predict the effectiveness of proposed countermeasures and evaluate their actual effectiveness after they are implemented in actual practice.

Vulnerability Analysis or Vulnerability Assessment involves following steps:

Defining and classifying network or system resources

Assigning relative levels of importance to the resources

Identifying potential threats to each resource

Developing a strategy to deal with the most serious potential problems

Defining and implementing ways to minimize the consequences if an attack occurs.

<<<<<< ==================== >>>>>>

Q. 114: What is a Vulnerability Scanner?

A vulnerability scanner is a tool for detecting & reporting genuine vulnerabilities in the system. It uses an up-to-date database containing complete information necessary to check a system for security holes. It provides facility to carry out multiple manual scans at a time. Its reports provide recommendations for countermeasures to remove the vulnerabilities detected by it.

<<<<<< ==================== >>>>>>

Q. 115: What is an Ethical Hacker?

Ethical hacker is a computer and network expert who legitimately attacks the security system on behalf of the management with an objective to find vulnerabilities, which any malicious hacker could exploit.

For testing a security system, ethical hackers use the same methods as malicious hackers, but their aim is to report back the problems instead of taking advantage of out of them. Ethical hacking is also known as penetration testing or intrusion testing.

<<<<<< ==================== >>>>>>

Q. 116: What is Database testing?

Database testing involves the following activities:

1) Testing of Data validity.

2) Testing of Data Integritity.

3) Performance testing related to the data base.

4) Testing of Procedure, triggers and functions.

<<<<<< ==================== >>>>>>

Q. 117: What are the things checked in Database Testing?

Following things are generally checked In Database Testing:

1) Validation of field size

2) Checking of constraints.

3) Checking of Indexes as to whether done or not

4) Checking of stored procedures

5) Checking as to whether the field size defined in the application is matching with that in the database or not

<<<<<< ==================== >>>>>>

Q. 118: What is High Order Testing?

High Order Testing is black-box testing conducted on the software after the completion of all integration activities.

<<<<<< ==================== >>>>>>

Q. 119: What is Internationalization or I18N?

Internationalization refers to the development and testing relating to handling foreign text and data within a software program in such a way that it will be easy to adapt it to several international markets having different languages and cultures. Internationalization includes sorting, importing and exporting text and data, correct handling of currency and date and time formats, string parsing and upper / lower case handling etc.

Method of deriving the abbreviation I18N for Internationalization goes like this:

First, we take the first letter of the word Internationalization we want to abbreviate; in this case the letter "I". Next, we take the last letter in the word; in this case the letter "N". These become the first and last letters in the abbreviation. Finally, we count the remaining letters in the word between the first and last letter. In this case. "nternationalizatio" has 18 characters in it. Thus we shall encapsulate the number 18 between the "I" and "N"; thus making the final abbreviation as I18N.

<<<<<< ==================== >>>>>>

Q. 120: What is localization or L10N?

Localization refers to development, testing and adapting the software product to suit a local or regional market. The objective of localization is to ensure suitability of the product with language & cultural aspects of the users of a particular region.

localization includes translating the program, choosing appropriate icons and graphics, and other cultural considerations. It also may include translating the program's help files and the documentation.

<<<<<< ========== >>>>>>

Q. 121: What is Globalization G11N?

Globalization refers to the activities performed for the purpose of marketing a software product in regional markets. The objective of globalization is to take care of global marketing accounting for economic and legal factors. The main focus of globalization is on marketing providing total enterprise solutions and a support to the management.

<<<<<< ========== >>>>>>

Q. 122: What are the benefits of Software Validation?

Software validation is an important tool employed to assure the quality of the software products. Few benefits are as under:

1) It increases the usability and reliability of the device software, resulting in reduced failure rates, less recalls and corrective actions, less liability to device manufacturers.

2) It reduces the long term costs by making it easier and less costly to reliably modify software and revalidate software changes.

3) It helps to reduce the long-term cost of software by reducing the cost of validation for each subsequent release of the software.

<<<<<< ========== >>>>>>

Q. 123: What is the role of Design Reviews in Software Development Life Cycle?

Design review is a primary tool for managing and evaluating software development projects. Design reviews allow management to confirm that all goals defined in the software validation plan have been achieved. Formal design reviews are more structured and include participation from others outside the development team.

Design reviews are documented, comprehensive, and systematic examinations of a design to evaluate the adequacy of the design requirements, to evaluate the capability of the design to meet these requirements, and to identify problems.

Design reviews include examination of development plans, requirements specifications, design specifications, testing plans and procedures, all other documents and activities associated with the project.

<<<<<< ========== >>>>>>

Q. 124: What is the need of Software Validation after a change?

When any change even a small one is made to the software, following activities need to be performed:

1) Re-establishment of the validation status of the software.

2) Conducting necessary validation analysis - not for the sake of validation of the individual change, but o to know the effect of the change on the entire software system.

3) Conducting suitable level of regression testing to show that unchanged but vulnerable portions of the system have not been adversely affected. Regression testing is meant to provide a confidence that the software has been validated after the change.

<<<<<< ========== >>>>>>

Q. 125: What is Output Forcing?

Output Forcing is a sort of functional testing of software applications.

It refers to choosing test inputs to ensure that all or the selected software outputs are generated by the testing.

<<<<<< ========== >>>>>>