DUMPSARENA

ISTQB Certified Tester Foundation Level (CTFL) v4.0

ISTQB ISTQB-CTFL

Version Demo

Total Demo Questions: 43

Total Premium Questions: 432

Buy Premium PDF

https://dumpsarena.co

sales@dumpsarena.co

sales@dumpsarena.co
dumpsarena.co

Topic Break Down

Topic	No. of Questions
Topic 1, Fundamentals of Testing	52
Topic 2, Testing Throughout the Software Development Lifecycle	74
Topic 3, Static Testing	44
Topic 4, Test Analysis and Design	129
Topic 5, Managing the Test Activities	112
Topic 6, Test Tools	21
Total	432

QUESTION NO: 1

Which of the following statements about independent testing is WRONG?

A. Independent testing is necessary because developers don't know any testing.

B. Independent testing is best suited for the system test level.

C. A certain degree of independence makes the tester more effective at finding defects.

D. Independent test teams may find other types of defects than developers who are familiar with the system's structure.

ANSWER: A

Explanation:

“Independent testing is necessary because developers don't know any testing.” is the wrong statement. Developers have an essential role in testing: they commonly design and execute tests during development, particularly at component level, and may also contribute to integration testing and other test activities. Their detailed knowledge of the implementation is valuable for identifying technical risks and checking whether the software behaves as intended.

Independence is not a substitute for developer testing, nor is it required because developers are supposedly unable to test. Instead, it is a complementary arrangement that can provide a different perspective. Testers with an appropriate degree of independence may be less influenced by assumptions made during implementation and can identify defects, risks, or quality concerns that might otherwise be overlooked. The appropriate degree of independence depends on the project context, including risk, regulation, organizational structure, and the test level involved.

The ISTQB Foundation Level syllabus describes independence as a way to obtain different perspectives while recognizing that developers also perform valuable testing activities. See the official ISTQB Certified Tester Foundation Level information and the ISTQB Glossary entry for independent testing.

QUESTION NO: 2

A team's test strategy was to invest equal effort in testing each of a system's modules. After running one test cycle, it turned out that most of the critical bugs were detected in one of the system's modules.

Which testing principal suggests a change to the current test strategy for the next test cycle?

A. Pesticide Paradox

B. Early testing

C. Absence-of-errors fallacy

D. Defect clustering

ANSWER: D

Explanation:

Defect clustering is the applicable testing principle because the test results show that critical defects are concentrated in one module rather than being distributed evenly throughout the system. This principle recognizes that a relatively small number of components often contain a disproportionately large number of defects or are responsible for most operational failures. The outcome of the first test cycle is therefore useful risk information: the module where most critical bugs were found has demonstrated a higher likelihood and impact of further defects than the other modules.

For the next test cycle, the team should use this evidence to adapt its strategy and allocate test effort according to risk, rather than continuing to give every module equal attention. This can include deeper testing of the affected module, increased coverage of its interfaces and complex logic, more exploratory testing, and focused regression testing after fixes. The ISTQB CTFL syllabus identifies defect clustering as one of the fundamental testing principles and notes that anticipated defect clusters can be identified during risk analysis and used to focus testing effort. See the ISTQB Certified Tester Foundation Level certification page and the ISTQB CTFL v4.0.1 syllabus.

QUESTION NO: 3

A test score indicator for students produces a performance score based on a combination of the number of consecutive hours studied (below 4 hours, 4 to 8 hours. 9 to 12 hours or above 12 hours) and the average intensity of focus on the material during the study time (low, medium or high).

Given the following test cases:

hours intensity score

T1 3 low55

T2 14 high 95

T3 9 low75

What is the minimum number of additional test cases that are needed to ensure full coverage of all valid INPUT equivalence partitions?

A. 1

B. 2

C. 3

D. 4

ANSWER: A

Explanation:

1 is correct because equivalence partitioning coverage for valid inputs requires that every valid equivalence partition for each input be exercised by at least one test case. The study-hours input has four valid partitions: below 4 hours, 4 through 8 hours, 9 through 12 hours, and above 12 hours. The existing tests already cover below 4 hours through the value 3, 9 through 12 hours through the value 9, and above 12 hours through the value 14. Therefore, the only uncovered valid hours partition is 4 through 8 hours. For focus intensity, the existing tests cover low and high, leaving medium as the only uncovered valid partition. A single additional test can cover both remaining partitions simultaneously, for example, using 6 study hours with medium focus intensity. The resulting score is not relevant to the requested coverage because the question specifically asks for valid *input* equivalence partitions. This application of equivalence partitioning is consistent with the CTFL syllabus's black-box test-technique guidance and ISTQB's glossary definition of the technique. See the ISTQB CTFL certification page and the ISTQB glossary entry for equivalence partitioning.

QUESTION NO: 4

Which of the following statements about static testing and dynamic testing is true?

A. Unlike dynamic testing, which can be also performed manually, static testing cannot be performed without specialized tools

B. Static testing is usually much less cost-effective than dynamic testing

C. Unlike dynamic testing, which focuses on detecting potential defects, static testing focuses on detecting failures which may be due to actual defects

D. Both static testing and dynamic testing can be used to highlight issues associated with non-functional characteristics

ANSWER: D

Explanation:

Both static testing and dynamic testing can be used to highlight issues associated with non-functional characteristics. Static testing examines work products without executing the software. For example, reviews and static analysis can reveal ambiguity, inconsistency, incompleteness, maintainability concerns, security weaknesses, or performance-related risks in

requirements, architecture, designs, code, and testware. Identifying such concerns early supports the prevention of defects and enables teams to improve quality attributes before the affected software is run.

Dynamic testing involves executing the software and observing its behavior. It can therefore provide evidence about non-functional characteristics in operation, such as response time and resource utilization for performance, resistance to unauthorized access for security, recoverability for reliability, and ease of use for usability. The two forms of testing are complementary: static testing is valuable for finding issues in work products early, while dynamic testing evaluates behavior of the executing system. The ISTQB CTFL syllabus explicitly recognizes static testing as applicable to various work products and as a means of detecting defects directly, while dynamic testing executes software to identify failures and assess quality characteristics. See the ISTQB Certified Tester Foundation Level certification page and the ISTQB Glossary for the relevant terminology.

QUESTION NO: 5

When should component integration tests be carried out?

A. Integration tests should always be done after system tests

B. Integration tests should be done at the customer's site, after acceptance tests

C. Integration tests can be done before or after system tests

D. Integration tests should always be done before system tests

ANSWER: D

Explanation:

“Integration tests should always be done before system tests” is the correct answer in the context of component integration testing and the CTFL test-process model. Component integration testing focuses on the interfaces, communications, and interactions among individual components after those components have been tested separately. Its purpose is to detect defects such as incorrect data passed between modules, interface mismatches, improper sequencing, and failures in component collaboration before the complete system is evaluated as a whole.

The CTFL syllabus describes test levels as activities that are usually performed sequentially: component testing is followed by component integration testing, and then system testing. Performing component integration testing before system testing helps isolate integration defects at the appropriate level, where diagnosis and correction are generally more focused and manageable. Once the integrated components have demonstrated the intended behavior, system testing can assess the behavior of the complete system against its specified requirements and end-to-end workflows. This ordering supports early feedback and reduces the risk that lower-level integration failures obscure broader system-test results. See the ISTQB Certified Tester Foundation Level certification page and the ISTQB glossary entry for integration testing.

QUESTION NO: 6

During component testing of a program if 100% decision coverage is achieved, which of the following coverage criteria is also guaranteed to be 100%?

A. 100% Stale transition coverage

B. 100% Equivalence class coverage

C. 100% Boundary value coverage

D. 100% Statement coverage

ANSWER: D

Explanation:

100% Statement coverage is guaranteed because decision coverage subsumes statement coverage. Decision coverage measures whether every possible decision outcome in the code has been exercised by the test suite. For a two-way

decision, such as an `if` statement, this means that tests execute both the true outcome and the false outcome. Exercising all decision outcomes causes execution to enter the code associated with each reachable branch. Consequently, every executable statement that is reachable through those outcomes will have been executed at least once, satisfying statement coverage. The ISTQB CTFL syllabus explicitly identifies this hierarchy: achieving 100% decision coverage guarantees 100% statement coverage. This relationship is useful when selecting structural coverage targets during component testing, because decision coverage provides stronger evidence of control-flow exercise than statement coverage alone. Coverage measurements should still be interpreted alongside the test basis, risk assessment, and the quality of test oracles; a high structural-coverage percentage demonstrates execution of code elements but does not by itself establish that all relevant behaviors have been fully verified. See the ISTQB Certified Tester Foundation Level certification materials and the ISTQB website for the official syllabus and terminology.

QUESTION NO: 7

A requirement states: “A customer may cancel an order before it has been dispatched.” During a review, a tester asks whether a cancellation must be rejected after dispatch and whether the customer should receive a confirmation message when a cancellation succeeds.

Which ONE of the following is the MAIN benefit of raising these questions at this stage?

A. They establish complete statement coverage for the cancellation function.

B. They identify unclear and missing behaviour before it can propagate into later work products.

C. They confirm that the cancellation function has already been implemented correctly.

D. They remove the need to perform dynamic testing of order cancellation.

ANSWER: B

Explanation:

The questions expose ambiguities and missing acceptance criteria before implementation starts. Static testing of requirements can identify defects such as omissions, inconsistencies, and non-testable statements early, when they are normally less costly to resolve. Designing tests alone would not remove the need to clarify the intended behaviour with the relevant stakeholders.

QUESTION NO: 8

Which of the following statements about TDD, BDD and ATDD is TRUE?

A. Refactoring is a practice that is an integral part of TDD and is applied both to tests and to code written to satisfy those tests.

B. ATDD is a black-box test design technique that is applicable exclusively at acceptance test level.

C. BDD is a developer practice where business stakeholders are not usually involved as the tests are directly written at unit/component test level.

D. ATDD is the practice of running the automated acceptance tests as part of a continuous integration process.

ANSWER: A

Explanation:

“Refactoring is a practice that is an integral part of TDD and is applied both to tests and to code written to satisfy those tests.” is correct because test-driven development follows a short, repeating development cycle: first create a test for the intended behavior, then implement the minimum production code needed to make that test pass, and then improve the internal structure while preserving behavior. This final activity is refactoring. It is not limited to production code: test code is also software that needs to remain clear, expressive, maintainable, and free of unnecessary duplication as the test suite grows. Refactoring tests can improve their names, fixtures, helpers, and structure without changing the behavior they verify.

Refactoring production code similarly improves design and readability without changing externally observable behavior, with the automated tests providing rapid feedback that behavior has been preserved. The CTFL syllabus identifies TDD as a test-first approach involving tests, implementation, and refactoring, and emphasizes its role in supporting maintainable code. See the ISTQB Certified Tester Foundation Level page for the official CTFL syllabus and supporting materials, and the ISTQB Glossary entry for test-driven development for the associated terminology.

QUESTION NO: 9

Which of the following is correct with regards to debugging?

A. Debugging identifies the cause of a failure

B. Debugging is often performed by test engineers

C. Debugging is considered part of the testing activities

D. Debugging is intended to find as many defects as possible in the code

ANSWER: A

Explanation:

“Debugging identifies the cause of a failure” is correct. In the ISTQB glossary, debugging is defined as the process of finding, analyzing, and removing the causes of failures in software. A failure is an observed deviation of the component or system from its expected behavior. Debugging begins after a failure has been observed or reported and focuses on investigating the underlying technical cause, such as an error in code, configuration, data handling, or an interface. Once the cause has been identified, debugging commonly includes correcting the defect and checking that the correction resolves the failure without introducing unintended effects. This work supports testing, but it is a distinct activity: testing provides information about product quality and can reveal failures, whereas debugging diagnoses and removes their causes. ISTQB also recognizes that debugging is normally performed by developers rather than being a testing activity itself. The distinction is important for clear responsibilities and effective collaboration: testers report reproducible failure evidence, while the people responsible for the implementation investigate and fix the underlying defect. See the ISTQB Glossary entry for debugging and the ISTQB Certified Tester Foundation Level syllabus resources.

QUESTION NO: 10

Which of the following statements best describes how configuration management supports testing?

A. Configuration management helps reduce testing effort by identifying a manageable number of test environment configurations in which to test the software, out of all possible configurations of the environment in which the software will be released

B. Configuration management is an administrative discipline that includes change control, which is the process of controlling the changes to identified items referred to as Configuration Items '

C. Configuration management is an approach to interoperability testing where tests are executed in the cloud, as the cloud can provide cost-effective access to multiple configurations of the test environments

D. Configuration management helps ensure that all relevant project documentation and software items are uniquely identified in all their versions and therefore can be unambiguously referenced in test documentation

ANSWER: D

Explanation:

Configuration management helps ensure that all relevant project documentation and software items are uniquely identified in all their versions and therefore can be unambiguously referenced in test documentation. This directly supports testing because testers need to know precisely which version of a requirement, test basis, testware item, build, environment definition, and software component is being used. Unique identification and version control create reliable links between test conditions, test cases, test results, defects, and the specific configuration of the test item on which testing was performed.

This makes results repeatable and auditable, supports impact analysis when changes occur, and enables the team to reproduce failures using the same controlled configuration. The CTFL syllabus identifies configuration management as a key supporting process that establishes and maintains the integrity of testware, the test object, and their relationships. It therefore provides the traceability and consistency needed for effective test monitoring, reporting, defect management, and release decisions. The ISTQB glossary likewise defines configuration management around identifying configuration items and controlling changes to them. See the ISTQB Certified Tester Foundation Level materials and the ISTQB Glossary entry for configuration management.

QUESTION NO: 11

Given the following statements:

1.It can prevent defects by manual examination of the functional specification

2.It is effective since it can be performed very early in the software development life cycle

3.It can detect the failures in the running application

4.It can help eliminate defects in user stories

5.It can verify externally visible behaviors

Which set of statements represent values ONLY for static testing?

A. 1,3, 4.5

B. 2,4,5

C. 1,2,4

D. 1,2, 3. 4,5

ANSWER: C

Explanation:

1,2,4 is correct because these statements describe recognized benefits of static testing. Static testing evaluates work products without executing software. For example, testers and other stakeholders can manually examine a functional specification through a review, identify ambiguities, omissions, inconsistencies, and other defect patterns, and have them corrected before those issues propagate into design, code, or tests. This early feedback is a central value of static testing: reviews can begin as soon as a work product is available, so they support defect prevention and reduce the cost and effort associated with later rework.

User stories are also valid static-test work products. Reviewing a user story collaboratively can expose unclear acceptance criteria, missing business rules, contradictions, or requirements that are not testable. Resolving such issues improves the quality of the story before implementation starts and helps the team build a shared understanding of the intended behavior. Thus, manual examination of the functional specification, early lifecycle application, and eliminating defects in user stories are all values specific to static testing. The CTFL syllabus describes static testing as examining work products and emphasizes early defect detection and prevention. See the ISTQB Certified Tester Foundation Level certification page and the CTFL v4.0.1 syllabus.

QUESTION NO: 12

In what way do Configuration Management effects testing?

A. Without proper configuration management, test planning cannot proceed.

B. Proper configuration management ensures that testers can uniquely identity the tested item

C. Configuration management is important for developers, not for testers

D. There is very little influence of configuration management practices on the test project.

ANSWER: B

Explanation:

Proper configuration management ensures that testers can uniquely identity the tested item. Configuration management establishes and maintains the integrity of work products throughout their life cycle by identifying configuration items, controlling changes, recording their status, and making the appropriate versions available. In testing, this means that a tester can determine exactly which version, build, component set, or environment was tested. That unambiguous identification is essential when recording test results and defects, because a result is meaningful only when it can be associated with a specific test object and its configuration. It also supports repeatability: another tester can recreate the same setup and rerun the tests against the same identified item. The CTFL syllabus describes configuration management as supporting test planning, test monitoring and control, incident management, and the maintenance of traceability among testware and test items. It applies to both the test object and testware, including test cases, test data, test environments, and test results. Therefore, uniquely identifying the tested item is a direct and fundamental testing benefit of proper configuration management. See the ISTQB Certified Tester Foundation Level certification page and the ISTQB glossary entry for configuration management.

QUESTION NO: 13

For the same financial institution in Question 12, with the same requirements and expectations, what would be the most likely investment values used in testing if two-point boundary value analysis is used to design test cases specific to the 13% interest rate equivalence partition?

A. R100 000, R100 001, R500 000, R500 001.

B. R99 999, R100 000, R499 999, R500 000.

C. R100 000. R500 000.

D. R99 000, R500 001.

ANSWER: A

Explanation:

R100 000, R100 001, R500 000, R500 001. is correct because two-point boundary value analysis selects two test values for each boundary of the relevant equivalence partition: the boundary value itself and the value immediately on the adjacent side of that boundary. Based on the stated 13% interest-rate partition, the lower edge is crossed between R100 000 and R100 001. Testing these adjacent values checks the transition from the preceding investment band into the band that attracts 13% interest. The upper edge is crossed between R500 000 and R500 001. These values similarly check the transition from the 13% band into the following investment band. Together, the four values exercise both boundaries and the behavior immediately on either side of each decision point. This is the intended use of two-point boundary value analysis: focus test design on defects that commonly occur where a numeric rule changes, such as an inclusive versus exclusive comparison or an incorrectly implemented threshold. The ISTQB Foundation Level syllabus describes boundary value analysis as deriving tests from the boundaries of equivalence partitions and distinguishes the two-value approach from the three-value approach. See the ISTQB Certified Tester Foundation Level page and the ISTQB glossary definition of boundary value analysis.

QUESTION NO: 14

Which of the following statements is true?

Select exactly one option (1 out of 4)!

A. Errors are the inevitable result of software failure.

B. Defects in the program code may be caused by environmental conditions.

C. Defects in the program code may result in failures in the program during execution.

D. Any deviation of the actual from the expected result represents an error.

ANSWER: C

Explanation:

Defects in the program code may result in failures in the program during execution. This correctly reflects the ISTQB distinction and causal relationship between an error, a defect, and a failure. A person can make an error, such as a misunderstanding while specifying, designing, coding, configuring, or maintaining software. That error may introduce a defect (also called a bug or fault) into a work product, including program code. When the defective code is executed, it may cause the software to behave differently from its specified or expected behavior; this observed incorrect behavior is a failure. The word “may” is important: the presence of a defect does not guarantee that a failure will occur. The relevant execution path might not be exercised, or particular data, timing, configuration, or environmental conditions may be required for the defect to trigger. Understanding this chain helps testers design tests that expose defects through observable failures, while recognizing that testing demonstrates the presence of defects rather than proving their complete absence. The terminology and relationship are defined in the ISTQB Certified Tester Foundation Level syllabus and glossary: ISTQB CTFL certification and syllabus resources and ISTQB Glossary.

QUESTION NO: 15

Which of the following lists factors That contribute to PROJECT risks?

A. skill and staff shortages; problems in defining the right requirements, contractual issues.

B. skill and staff shortages; software does not perform its intended functions; problems in defining the right requirements.

C. problems in defining the right requirements; contractual issues; poor software quality characteristics.

D. poor software quality characteristics; software does not perform its intended functions.

ANSWER: A

Explanation:

The correct list is “skill and staff shortages; problems in defining the right requirements, contractual issues.” ISTQB defines project risks as risks that can affect the project’s ability to achieve its objectives, such as delivering the agreed scope within planned time, cost, and resource constraints. These risks arise from conditions surrounding the project and its management rather than from a potential defect or quality characteristic of the delivered product.

Skill and staff shortages are organizational risk factors because insufficient availability of appropriately skilled people can affect estimates, planned work, test activities, communication, and delivery dates. Problems in defining the right requirements are technical project-risk factors: unclear, incomplete, inconsistent, or unsuitable requirements make it difficult to plan, build, and test the intended solution effectively. Contractual issues are supplier-related project-risk factors, since agreements with customers, suppliers, or external parties can create delivery, responsibility, dependency, or acceptance risks. Together, these factors directly represent recognized sources of project risk in the CTFL syllabus’s treatment of risks and testing.

See the official ISTQB Certified Tester Foundation Level certification page and the ISTQB official website for the current CTFL syllabus and supporting certification materials.

QUESTION NO: 16

Which of the following statements regarding the test-first approach (principle of early testing) is true?

A. An approach where the tests are written only as needed.

B. An approach where the tests are written after implementation.

C. An approach where the tests are written during implementation.

D. An approach where the tests are written before implementation.

ANSWER: D

Explanation:

The correct statement is “An approach where the tests are written before implementation.” A test-first approach deliberately creates tests before the corresponding code is implemented. This lets the tests express the intended behavior, acceptance criteria, or detailed design of the item under development before implementation begins. Developers then implement only enough code to make the initially failing test pass, subsequently improving the design while retaining the tests as an automated regression safety net. Test-driven development is a common example of this approach: a test is created first, the implementation is added to satisfy it, and the code may then be refactored. This embodies early testing because test design and test execution planning begin as early as possible in the software development lifecycle, enabling defects in requirements, interfaces, and expected behavior to be identified sooner. Early feedback also improves collaboration, because the expected outcome is made concrete and reviewable before code is completed. The ISTQB CTFL syllabus describes test-first approaches as writing tests before the code is written and identifies test-driven development as an example. See the ISTQB Certified Tester Foundation Level certification page and the ISTQB CTFL Syllabus v4.0.1.

QUESTION NO: 17

In which one of the following test techniques are test cases derived from the analysis of the software architecture?

A. Black-box test techniques.

B. Experience-based test techniques.

C. Checklist-based test techniques.

D. White-box test techniques.

ANSWER: D

Explanation:

White-box test techniques are correct because they derive test cases from the internal structure of the test object. That structure can include the software’s code, control flow, data flow, and architecture. When architecture is used as the test basis, a tester can design tests to exercise structural elements and their relationships, such as components, interfaces, branches, paths, or interactions represented by the implementation design. This approach requires knowledge of how the system is built, rather than relying solely on externally observable behavior or personal testing experience. In the ISTQB CTFL syllabus, white-box testing is defined as testing based on the analysis of the internal structure of a component or system. The syllabus also identifies coverage as an important measure for white-box techniques: coverage indicates the extent to which selected structural elements have been exercised by the test suite. Therefore, analyzing software architecture to derive tests is a structural, white-box testing activity. See the ISTQB Certified Tester Foundation Level certification page and the ISTQB CTFL Syllabus v4.0.1, section on white-box test techniques.

QUESTION NO: 18

A financial institution is to implement a system that calculates the interest rates paid on investment accounts based on the sum invested.

You are responsible for testing the system and decide to use equivalence partitioning and boundary value analysis to design test cases. The requirements describe the following expectations:

Investment range| Interest rate

R500 to RIO 00010%

RIO 001 to R50 00011%

R50 001 to RIOOOOO12%

RIOOOOl to R500 000| 13%

What is the minimum number of test cases required to cover all valid equivalence partitions for calculating the interest?

A. 5

B. 4

C. 8

D. 16

ANSWER: B

Explanation:

The correct answer is **4**. Equivalence partitioning divides the investment-input domain into groups for which the system is expected to behave equivalently. Here, each stated investment range has a distinct expected interest rate, so each range is a separate valid equivalence partition: R500–R10,000, R10,001–R50,000, R50,001–R100,000, and R100,001–R500,000. To achieve coverage of all valid equivalence partitions, the test suite needs at least one representative investment amount from each of these four ranges. Therefore, four test cases are the minimum required for the coverage requested. For example, suitable representative values could be R500, R10,001, R50,001, and R100,001, although the particular selected values are not important for equivalence-partition coverage as long as each lies within its respective valid range. Boundary value analysis may also be used when designing additional tests around the range limits, but it does not change the minimum when the question specifically asks for coverage of all valid equivalence partitions. The ISTQB CTFL syllabus describes equivalence partitioning as dividing data into partitions that can be treated in the same way, with tests designed to cover those partitions. See the ISTQB Certified Tester Foundation Level overview and the ISTQB CTFL v4.0 syllabus.

QUESTION NO: 19

Which of the following about typical information found within a test plan is FALSE?

A. The need to temporarily have additional test personnel available for specific test phases and/or test activities

B. The conditions that must be met in order for the test execution activities to be considered completed.

C. The list of the product risks which have not been fully mitigated at the end of test execution.

D. The conditions that must be met for part of all the planned activities to be suspended and resumed.

ANSWER: C

Explanation:

The list of the product risks which have not been fully mitigated at the end of test execution is the false statement because this information is normally recorded and communicated in the test completion report, rather than being typical planned content of a test plan. A test plan is created to guide the intended test activities: it establishes the scope and objectives of testing, approach, schedule, responsibilities, resource needs, and the criteria used to control and complete the work. In contrast, the residual product risks remaining after test execution are an outcome of testing. They must be assessed, documented, and provided to stakeholders so that an informed release or deployment decision can be made. This information reflects the actual status at the end of the testing effort, including risks that remain due to incomplete coverage, unresolved defects, or other limitations. The ISTQB CTFL syllabus distinguishes planning information from test completion reporting and identifies residual risks as important completion-report content. See the ISTQB Certified Tester Foundation Level certification page and the ISTQB CTFL v4.0.1 syllabus.

QUESTION NO: 20

A state transition diagram describes a control system's behavior in different operational modes. The initial state is **"NORMAL MODE"**.

Which ONE of the following test cases covers an **INVALID sequence**?

A. NORMAL MODE → DIAGNOSTIC MODE → DEGRADED MODE → EMERGENCY MODE → DIAGNOSTIC MODE → NORMAL MODE → DIAGNOSTIC MODE

B. NORMAL MODE → DEGRADED MODE → NORMAL MODE → DIAGNOSTIC MODE → DEGRADED MODE → EMERGENCY MODE → DIAGNOSTIC MODE

C. NORMAL MODE → DIAGNOSTIC MODE → DEGRADED MODE → EMERGENCY MODE → DIAGNOSTIC MODE → NORMAL MODE → DEGRADED MODE

D. NORMAL MODE → DIAGNOSTIC MODE → NORMAL MODE → DIAGNOSTIC MODE → EMERGENCY MODE → DIAGNOSTIC MODE → NORMAL MODE

ANSWER: D

Explanation:

NORMAL MODE → DIAGNOSTIC MODE → NORMAL MODE → DIAGNOSTIC MODE → EMERGENCY MODE → DIAGNOSTIC MODE → NORMAL MODE is the invalid sequence because it includes a transition directly from **DIAGNOSTIC MODE** to **EMERGENCY MODE**. Based on the state model implied by the listed operational behavior, DIAGNOSTIC MODE may return to NORMAL MODE or progress to DEGRADED MODE, while entry to EMERGENCY MODE occurs from DEGRADED MODE. The direct DIAGNOSTIC MODE-to-EMERGENCY MODE step therefore attempts a state change that is not defined by the model.

State transition testing derives tests from permitted states and transitions in a state-transition diagram or table. A valid transition test confirms that the system can follow a specified transition when its triggering condition occurs. Conversely, an invalid-transition test deliberately attempts a transition that is absent from the model and verifies that the system handles the attempt appropriately, for example by rejecting it, remaining in its current state, or reporting an error according to the requirements. This sequence reaches DIAGNOSTIC MODE through valid setup steps and then exercises the undefined transition, making it an appropriate invalid-sequence test. ISTQB identifies state transition testing as a black-box technique applicable where system behavior depends on current and previous states; see the ISTQB CTFL certification page and the CTFL syllabus resources.

QUESTION NO: 21

Given the following examples of entry and exit criteria:

1.A defined level of code coverage has been achieved

2.The test automation tool has been installed and properly configured

3.The number of unresolved defects is within the predefined limit

4.The performance test environment has been set-up and is available

5.The user stories have proper acceptance criteria defined

6.The testing budget has been spent and the project sponsor bears the risk of not testing any further

Which of the following BEST categorizes them as entry and exit criteria:

A. Entry criteria – 2, 4, 5; Exit criteria – 1, 3, 6

B. Entry criteria - 2, 4 Entry criteria - 2, 4, 5, 6

C. Exit criteria -1,3,6Exit criteria - 2, 5. 6

D. Exit criteria -1,3,56Exit criteria -1,3

ANSWER: A

Explanation:

“Entry criteria – 2, 4, 5; Exit criteria – 1, 3, 6” is correct because entry criteria define the preconditions that should be fulfilled before a test activity can begin. A configured test automation tool provides the required test capability, an available performance test environment provides the required infrastructure, and user stories with defined acceptance criteria provide a sufficiently testable basis for designing and executing tests. These conditions help ensure that testing starts with suitable resources, environments, and testable requirements in place.

Exit criteria define the conditions used to decide whether a test activity can be considered complete or should be stopped. Achieving a specified code-coverage level is a measurable completion target. Having unresolved defects within an agreed limit demonstrates that the remaining product risk is within the defined tolerance. Where the testing budget has been exhausted and the sponsor explicitly accepts the risk of no further testing, this is also a valid basis for stopping, provided the decision and accepted risk are communicated and managed appropriately. The CTFL syllabus describes entry and exit criteria as part of test planning and control, supporting informed decisions on when testing can start and when it can finish. See the ISTQB Certified Tester Foundation Level and the CTFL v4.0 syllabus resources.

QUESTION NO: 22

Consider the following **iteration planning tasks** where a tester can provide value:

Break down user stories into tasks (particularly testing tasks)

Estimate test effort for all testing tasks

Identify and refine functional and non-functional aspects of the test object

Which ONE of the following tasks should be ADDED to the above list?

A. Determining the test strategy

B. Participating in the detailed risk analysis of user stories and determining their testability

C. Planning the testing for the release

D. Writing testable user stories and acceptance criteria

ANSWER: B

Explanation:

Participating in the detailed risk analysis of user stories and determining their testability is the appropriate additional activity because iteration planning is the point at which the team turns selected user stories into actionable, estimable work for the upcoming iteration. A tester contributes specialist quality and risk knowledge while the stories are being discussed. Detailed risk analysis helps the team identify characteristics that may require particular test approaches, additional environments or data, specialist skills, or more effort. Determining testability establishes whether the intended functionality and non-functional behavior can be effectively observed, controlled, and evaluated, including whether suitable interfaces, logs, test data, and acceptance criteria are available. These findings directly support the existing planning activities: decomposing testing work into tasks, producing realistic test estimates, and refining the test object’s functional and non-functional aspects. The ISTQB CTFL syllabus specifically identifies detailed risk analysis of user stories and determining their testability among ways testers add value during iteration planning. This early collaboration allows risks and test constraints to influence the iteration plan before implementation begins, helping the team make a feasible commitment and build quality into the work. See the ISTQB Certified Tester Foundation Level certification page and the ISTQB website for the current CTFL syllabus and supporting materials.

QUESTION NO: 23

Which of the following statements about test reports are TRUE?

I Test reports shall be approved by the test team.

II. Test reports shall give stakeholders information as basis for decisions.

III Test reports shall summarize what happened through a period of testing.

IV. Test reports shall be approved by the development team, the test team and the customer

V. Test reports shall include information about remaining risks.

A. II, III, V

B. I, II, IV

C. I, III, v

D. II, III, IV

ANSWER: A

Explanation:

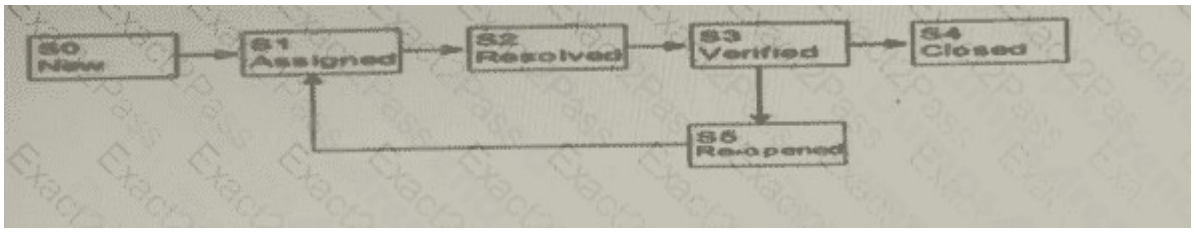
“II, III, V” is correct. In the ISTQB CTFL v4.0 syllabus, test reporting is used to communicate testing information to stakeholders so they can make informed decisions. The report content is tailored to its audience and purpose, but its decision-support role is central: stakeholders may use the reported information when deciding whether to continue testing, release a product, allocate resources, or accept residual quality exposure.

Test reports also provide a summary of a defined testing period, activity, or test level. This summary can cover the work performed and its outcomes, including progress against the test plan, executed tests, detected defects, achieved coverage, deviations, and relevant observations. By consolidating these facts, the report gives an understandable view of what occurred during testing rather than merely presenting isolated test results.

Remaining risks are an important part of completion-oriented reporting because they communicate unresolved exposure at the time of reporting. Such risks may arise from untested areas, known defects, environmental constraints, incomplete coverage, or uncertainty about product behavior. Reporting them enables stakeholders to assess whether the residual risk is acceptable. See the ISTQB Certified Tester Foundation Level page and the ISTQB CTFL v4.0.1 syllabus.

QUESTION NO: 24

Which sequence of state transition stated in the answer choices is correct in accordance with the following figure depicting me life-cycle of a defect?



A. S0- > S1- > S2- > S3- > S4

B. S0- > S1- > S2- > S3- > S5^ > S1

C. S0- > S1- > S2- > S3- > S5- > S1- > S2- > S3

D. S0- > S1- > S2- > S3- > S5- > S3- > S4

ANSWER: C

Explanation:

The sequence **S0- > S1- > S2- > S3- > S5- > S1- > S2- > S3** is correct because every successive pair of states follows a permitted directed transition in the depicted defect life-cycle. It begins with the defect moving from its initial state to the reported/registered state, then to the state in which it is assigned and fixed, and then to verification by testing. From verification, the model permits transition to S5 when testing does not confirm that the defect has been fixed. The defect can then be returned from S5 to S1 so that it re-enters the reporting/assignment flow. It is subsequently assigned and fixed again, returning to S3 for a further verification attempt. This is a valid state-transition path because it respects both the

direction of each transition and the rework loop represented by the model. State transition testing uses such a model to represent discrete system states and the events or conditions that cause movement between them; valid test paths must follow only transitions defined by the state diagram. See the ISTQB Certified Tester Foundation Level certification information and the ISTQB Glossary entry for state transition testing.

QUESTION NO: 25

A test engineer finds a defect while testing. After the developer has fixed the defect, the test engineer decides to re-run a complete section of the tests. Which of the following is correct?

A. The lest engineer should not re-run the tests, as they have already been run, and results recorded.

B. The test engineer should not re-run the tests, they should be part of the developer tests.

C. The test engineer should re-run the tests, in order to ensure that new defects have not been introduced by the fix.

D. The test engineer should re-run the tests, because the defect shows that the test cases need to be updated.

ANSWER: C

Explanation:

The test engineer should re-run the tests, in order to ensure that new defects have not been introduced by the fix. This is regression testing: testing of a previously tested component or system following change, to confirm that existing functionality still works as intended. Although the developer's change is intended to resolve the reported defect, modifications can have unintended effects on code, interfaces, data handling, shared components, or related behavior. Re-running a complete relevant section of tests therefore provides evidence that the fix has not adversely affected functionality that had worked before the change. The scope of regression testing should be selected using an impact assessment and risk considerations; in this situation, the complete section is the chosen scope. Regression testing is distinct from confirmation testing, which specifically checks that the originally reported defect has been corrected. Both may be performed after a defect fix, but the stated purpose—detecting defects newly introduced by the change—is explicitly regression testing. This practice supports maintaining confidence in product quality as software evolves through fixes and other modifications. The ISTQB CTFL syllabus defines regression testing in the context of changes and emphasizes its role in identifying unintended side effects. See the ISTQB Certified Tester Foundation Level certification page and the ISTQB Glossary definition of regression testing.

QUESTION NO: 26

Use Scenario 1 "Happy Tomatoes" (from the previous question).

When running test case **TC_59** , the actual result for **t = 35** degrees Celsius is **OUTPUT = X** instead of the expected output.

Which information should NOT be included in the defect report?

A. Identification of the test object and test environment

B. A concise title and a short summary of the defect being reported

C. Description of the structure of the test team

D. Expected results and actual results

ANSWER: C

Explanation:

Description of the structure of the test team is the information that should not be included in this defect report. The purpose of a defect report is to communicate a specific observed anomaly in sufficient detail for the relevant stakeholders to assess, reproduce, investigate, correct, and track it. Its contents should therefore focus on the failed test execution and the affected product: for example, the test object and environment, a concise defect summary, the steps or test case used, and the expected and actual results. In this scenario, recording that TC_59 was run with t = 35 degrees Celsius and produced OUTPUT = X provides evidence that supports analysis and repeatable reproduction of the problem. Team organizational

structure does not describe the product failure, its conditions, or its observable impact, so it does not help establish or resolve the reported discrepancy. The ISTQB CTFL syllabus identifies defect-report information such as identifiers, summary, test object and environment, expected and actual results, and lifecycle status; these fields support effective defect management. See the ISTQB Certified Tester Foundation Level certification page and the ISTQB certifications and supporting documents page.

QUESTION NO: 27

For a mandatory input field "ZIP code" the following rules are given:

1 - The valid ZIP code format is 5 numeric digits.

2 - The code has to exist in the post office's official ZIP code list

Using equivalence classes partitioning, how many test cases are required to test this field?

A. 8

B. 3

C. 6

D. 4

ANSWER: D

Explanation:

4 is correct because equivalence partitioning groups inputs that should be processed in the same way into representative classes. For this field, one representative test is needed for a valid, populated value that consists of five numeric digits and is present in the official ZIP-code list. Three invalid classes also need coverage: an omitted value, because the field is mandatory; a populated value that does not meet the required five-numeric-digit format; and a value that has the valid five-digit numeric format but is not in the official list. A single representative from each of these classes provides four test cases in total.

This applies the CTFL principle that, once equivalence partitions have been identified, testing one representative value from each partition is normally sufficient, provided that all members of that partition are expected to be handled equivalently. The format rule and the business-list membership rule must be considered separately: a syntactically well-formed ZIP code still requires validation against the authoritative list. The required-field rule is likewise a distinct input condition requiring a representative empty or missing value. See the ISTQB Certified Tester Foundation Level certification page and the ISTQB glossary definition of equivalence partitioning.

QUESTION NO: 28

Out of the following. what is not needed to specify in defect report?

A. Test environment details

B. How to reproduce the defect

C. How to fix the defect

D. Severity and priority

ANSWER: C

Explanation:

“How to fix the defect” is correct because the primary purpose of a defect report is to communicate sufficient, factual information for stakeholders to understand, reproduce, assess, track, and resolve a suspected anomaly. Its content should describe what was observed, what was expected, the conditions under which the issue occurred, and information supporting

impact assessment and workflow management. The report therefore enables the responsible team to investigate the underlying cause and decide the appropriate corrective action.

A specific implementation solution is not normally required from the person reporting the defect. Determining the technical correction can require design knowledge, source-code analysis, architectural considerations, and assessment of possible side effects; it is generally decided during investigation and resolution by the appropriate development or maintenance personnel. A reporter may optionally provide a workaround or suggestion when useful, but a prescribed method of fixing the issue is not essential defect-report information. This distinction keeps reports objective and focused on reproducible evidence rather than assumptions about implementation. ISTQB describes defect reporting as communicating anomalies found during testing and identifies the information that supports analysis, prioritization, and lifecycle tracking. See the ISTQB CTFL certification and syllabus resources and the ISTQB Glossary definition of defect report.

QUESTION NO: 29

How can testing contribute to higher quality?

A. Testing helps to measure the quality of software.

B. Testing ensures that remaining defects are documented.

C. Testing removes errors in the software.

D. Testing eliminates the risk with software.

ANSWER: A

Explanation:

Testing helps to measure the quality of software because it provides objective information about the extent to which a component or system meets specified requirements and satisfies users' and other stakeholders' needs. Testers evaluate the system by executing test cases or otherwise examining test items, observing actual results, and comparing those results with expected results and defined quality criteria. The resulting evidence enables stakeholders to assess relevant quality characteristics, such as functional suitability, performance efficiency, usability, reliability, security, and compatibility. This information supports informed decisions about release readiness, residual risk, priorities, and necessary improvement actions. Testing also reveals defects and failures that may affect the product's quality, allowing the organization to address them and to improve its development and quality-management processes. Importantly, testing is therefore a quality-evaluation activity: it makes the current quality level and product risks more visible to the people responsible for decisions. The ISTQB CTFL syllabus identifies evaluating work products, verifying requirements, providing information for decision-making, and reducing risk as important test objectives. The ISTQB glossary also defines quality in terms of fulfilling stated and implied needs. See the ISTQB Certified Tester Foundation Level syllabus and the ISTQB Glossary definition of quality.

QUESTION NO: 30

Consider the following testing levels :

1) Component Testing

2) Integration Testing

3) System Testing

4) Acceptance Testing

Which of the following statements is true?

A. Integration and system testing are applicable when V-model is used.Component and acceptance testing are applicable when iterative development models are used.

B. All the testing levels are applicable to V-model for software development.Only acceptance testing is applicable for iterative models.

C. Acceptance testing is applicable for all software development models.Component and system testing are applicable only for the V-model.

D. All testing levels are applicable, independent of which software development life-cycle process (V-model. iterative, incremental) is used.

ANSWER: D

Explanation:

All testing levels are applicable, independent of which software development life-cycle process (V-model, iterative, incremental) is used. The ISTQB CTFL syllabus identifies component, component integration, system, system integration, and acceptance testing as testing levels distinguished by their test object, test basis, and objectives. These levels describe the scope at which testing is performed, rather than prescribing a particular development life-cycle model. Consequently, a team using a sequential V-model can plan and perform each relevant level, while a team using iterative or incremental development can perform the same levels repeatedly within iterations, releases, or increments. For example, a newly developed component may be tested at component level during an iteration, then tested with collaborating components and external systems at appropriate integration levels; the completed product or increment can subsequently undergo system and acceptance testing. The timing, frequency, and degree of formality may differ according to the chosen life cycle, project risks, and delivery approach, but the applicability of a testing level is not limited to one development model. This principle supports selecting testing activities according to the product and its quality risks, while integrating them appropriately into the project's delivery process. See the ISTQB Certified Tester Foundation Level certification page and the ISTQB CTFL v4.0.1 syllabus.

QUESTION NO: 31

Given the following review types and review characteristics:

a.Pair review

b.Walkthrough

c.Technical review

d.Inspection

1.Formal

2.Informal

3.Purposes include evaluating the quality of the work product under review and generating new ideas (e.g., brainstorming solutions)

4.Purposes include Improving the software product and training the review participants

Which of the following BEST matches the review type with the review characteristic?

A. a-1. b-4, c-3. d-2

B. a-4, b-3. c-2. d-1

C. a-2, b-3, c-4, d-1

D. a-2, b-4, c-3. d-1

ANSWER: D

Explanation:

The correct match is "a-2, b-4, c-3. d-1". A pair review is an informal review in which two people jointly examine a work product, often as part of collaborative development activities. A walkthrough is typically led by the author and may be used to improve the work product while also educating or training participants through discussion and shared understanding. This educational aspect is a recognized purpose of walkthroughs. A technical review focuses on evaluating the quality of a work

product and can also be used to generate new ideas, including possible solutions to identified issues. An inspection is the most formal review type, characterized by a defined process, assigned roles, entry and exit criteria, and documented results. These associations reflect the CTFL syllabus distinction between informal reviews, walkthroughs, technical reviews, and formal inspections. The official ISTQB CTFL certification page provides the current syllabus and supporting materials: ISTQB Certified Tester Foundation Level. Review types and their objectives are covered in the syllabus topic on static testing and reviews; see also ISTQB.

QUESTION NO: 32

A tester is given a charter to investigate the error handling of a mobile banking application. While testing, the tester learns how the application behaves, designs new tests based on the observed behaviour, and immediately executes those tests. Notes are recorded to support later reporting.

Which ONE of the following test techniques is being used?

A. Decision table testing

B. State transition testing

C. Exploratory testing

D. Checklist-based testing

ANSWER: C

Explanation:

Exploratory testing is an experience-based technique in which test design, test execution, and learning occur at the same time. A charter can provide direction and boundaries, while the tester adapts subsequent tests according to observations made during execution.

Decision table testing and state transition testing derive tests systematically from models. Checklist-based testing uses a predefined list of items to be checked rather than the stated simultaneous learning, design, and execution.

QUESTION NO: 33

Which ONE of the following options BEST describes the purpose of confirmation testing versus regression testing?

A. The purpose of confirmation testing is to confirm that the defect giving rise to a failure has been successfully fixed. The regression test aims to ensure that no defects have been introduced or discovered in unmodified areas of the software as a result of the changes made.

B. Confirmation testing ensures the entire system functions as expected, whereas regression testing focuses only on modified components.

C. Confirmation testing verifies all system requirements, while regression testing ensures that no additional test cases are needed.

D. Regression testing and confirmation testing are interchangeable and serve the same purpose.

ANSWER: A

Explanation:

The purpose of confirmation testing is to verify that the specific defect which caused a reported failure has been successfully corrected. After a developer applies a fix, the relevant test that previously exposed the failure is re-executed, typically using the same test conditions, to provide evidence that the intended correction works. This is also commonly called retesting. Its focus is therefore narrow and directly linked to the particular defect and its fix.

Regression testing has a broader protective purpose following a change, such as a defect fix, enhancement, configuration change, or environment change. It checks that changes have not caused unintended effects in parts of the software that

were previously working, including areas not directly modified. Regression tests may include tests related to the changed area as well as tests for connected or potentially affected functionality. The option stating that confirmation testing confirms the defect has been fixed and regression testing seeks to detect defects introduced in unmodified areas accurately captures this distinction. The CTFL syllabus explicitly identifies confirmation testing and regression testing as different change-related testing activities. See the ISTQB Certified Tester Foundation Level certification page and the ISTQB CTFL v4.0 syllabus.

QUESTION NO: 34

Which of the following activities is NOT a pan of the fundamental testing process?

A. Archiving automation code

B. Test status reporting

C. Test process improvement

D. Build release and maintenance

ANSWER: D

Explanation:

Build release and maintenance is not an activity of the fundamental testing process. In CTFL v4.0, testing is organized through activities such as test planning, test monitoring and control, test analysis, test design, test implementation, test execution, and test completion. These activities focus on defining the test approach, deriving and preparing tests, executing them, communicating progress and results, and closing the test work appropriately. Test completion includes preserving relevant testware and other information for later use, as well as identifying opportunities to improve the test process. Test status reporting supports test monitoring and control by communicating progress, risks, and deviations from the plan to relevant stakeholders. In contrast, building a release and maintaining it are software development, configuration management, release management, and operational lifecycle responsibilities. Testing provides information that can support release decisions, but it does not itself build, deploy, or maintain the product release. Therefore, the activity outside the fundamental testing process is build release and maintenance. The CTFL syllabus describes the test process and its activities in its testing fundamentals material: ISTQB Certified Tester Foundation Level. The official syllabus is available from the ISTQB CTFL certification page.

QUESTION NO: 35

A system has valid input numbers ranging between 1000 and 99999 (both inclusive). Which of the following inputs are a result of designing tests for all valid equivalence classes and their boundaries?

A. 999.1000.23232.99999.100000

B. 999.1000.50000.100000.100001

C. 999.100000

D. 1000,50000,99999

ANSWER: A

Explanation:

The correct input set is *999.1000.23232.99999.100000*. Equivalence partitioning divides the input domain into groups expected to be processed in the same way. Here, the valid equivalence class consists of all whole-number inputs from 1000 through 99999, inclusive. The value 23232 is a representative value from within that valid class, so it provides a test of normal valid processing without merely retesting an endpoint.

Boundary value analysis then focuses on the edges of that valid class because defects frequently occur in comparisons, limits, and inclusivity handling. The lower boundary is 1000 and the upper boundary is 99999. For two-value boundary value analysis, the immediately adjacent values outside the range are also selected: 999 just below the lower boundary and

100000 just above the upper boundary. Together, these values test entry into the valid range, valid endpoint treatment, and exit from the valid range, while the representative valid value covers the valid equivalence class itself.

This application is consistent with the CTFL syllabus treatment of equivalence partitioning and boundary value analysis as black-box test techniques. See the ISTQB Certified Tester Foundation Level page and the ISTQB glossary definition of boundary value analysis.

QUESTION NO: 36

Which ONE of the following options is **NOT** a benefit of **test automation** ?

A. Reduced test execution times

B. More objective assessment

C. Prevention of simple human errors

D. Eliminating completely the need for manual testing

ANSWER: D

Explanation:

Eliminating completely the need for manual testing is not a benefit of test automation because automation is intended to support and extend testing, not replace all human testing activities. In the CTFL syllabus, test automation is associated with benefits such as reducing repetitive manual work, increasing the consistency and repeatability of test execution, and providing faster feedback. These strengths are especially useful for stable regression checks, frequent builds, and test cases that must be run repeatedly with the same data and expected results.

However, effective testing still needs human judgment. Testers must interpret risks, select appropriate test approaches, assess product behavior in context, investigate unexpected outcomes, and adapt testing as new information emerges. Human-led activities are particularly important when exploring a product, evaluating usability and user experience, reviewing requirements, and designing tests for novel or changing behavior. Automated checks execute the instructions that have been implemented; they do not independently determine whether those instructions are sufficient or whether the product is valuable and usable for its intended users. Therefore, automation can reduce manual effort for suitable work, but it cannot completely eliminate the need for manual testing. This aligns with the Foundation Level syllabus and its treatment of automation as a testing support mechanism: ISTQB Certified Tester Foundation Level and ISTQB Glossary.

QUESTION NO: 37

Which of the following tools is most likely to detect defects in functions or methods in source code?

A. configuration management tool

B. unit test framework tool

C. test design tool

D. monitoring tool

ANSWER: B

Explanation:

The correct answer is **unit test framework tool**. Unit testing focuses on individually testable components, such as functions, methods, classes, or modules. A unit test framework provides a structured way for developers and testers to create automated tests, run those tests repeatedly, compare actual results with expected results through assertions, and report failures. If a function produces an incorrect value, handles a boundary condition improperly, or fails to handle an expected error condition, an automated unit test can expose that defect directly and close to its source in the code.

Examples include JUnit for Java, NUnit for .NET, and pytest for Python. These frameworks support fast feedback because tests can be executed whenever code changes, including as part of continuous integration. They may also be used with coverage tools to identify code that has not been exercised, helping teams improve the completeness of their unit-test suites. The ISTQB CTFL syllabus identifies unit test frameworks as development-support tools that facilitate testing at component level. See the ISTQB Certified Tester Foundation Level certification page and the JUnit User Guide for practical framework capabilities.

QUESTION NO: 38

Which of the following statements are true?

1. Early and frequent feedback helps to avoid requirements misunderstanding.

2. Early feedback allows teams to do more with less.

3. Early feedback allows the team to focus on the most Important features.

4. Early and frequent feedback clarifies customer feedback by applying static testing techniques

Select the correct answer:

A. 3

B. 2

C. 1

D. 4

E. 1, 2 and 3

ANSWER: E

Explanation:

1, 2 and 3 is correct. The CTFL syllabus describes feedback as a key contributor to successful software development and testing. Feedback is most valuable when it is obtained early and frequently, because it gives the team timely information on the product, requirements, and stakeholder needs. In particular, early feedback helps prevent or resolve misunderstandings about requirements before those misunderstandings propagate into design, implementation, and test artifacts. This reduces avoidable rework and supports efficient use of the team's available time and effort—doing more with less. Frequent feedback also enables the team to assess value and priorities continually, helping it concentrate its work on the features that matter most to users and other stakeholders. These benefits apply throughout the software development lifecycle and are especially important when requirements, priorities, or product risks evolve. The wording reflects the CTFL treatment of feedback as a mechanism for improving communication, guiding decisions, and supporting delivery of the highest-value product increments. See the official ISTQB Certified Tester Foundation Level certification page and the CTFL syllabus resources for the syllabus material on the value of feedback in testing.

QUESTION NO: 39

Which of the following statements is an example of testing contributing to higher quality?

A. A test leader writes a test summary report

B. A project manager asks to a test leader to estimate the test effort

C. A tester installs a test item in the test environment

D. A tester finds a bug which is resolved prior to release

ANSWER: D

Explanation:

A tester finds a bug which is resolved prior to release is an example of testing contributing to higher quality because testing provides information that enables defects to be identified, corrected, and confirmed before the product reaches its users. A defect is a flaw in a work product that may cause the system to fail to perform a required function. By detecting the defect before release, the tester makes it possible for the team to remove a source of potential failure from the delivered product.

This contribution is not limited to executing tests: testing supports quality by evaluating work products and supplying objective information to stakeholders so that they can make informed decisions. In this case, the information produced by testing directly leads to corrective action. Resolving the bug prior to release reduces the likelihood that users will encounter incorrect behaviour in operation and therefore improves the extent to which the product meets specified requirements and user needs. Confirmation after the fix also gives evidence that the corrective action has achieved the intended result.

This reflects the CTFL view that testing is a quality-related activity: it helps achieve quality objectives by detecting defects early and providing useful quality information, while recognizing that testing itself is not the sole means of building quality. See the ISTQB Certified Tester Foundation Level overview and the ISTQB Glossary for the relevant terminology.

QUESTION NO: 40

The following state diagram is given as basis for state transition testing and contains only valid transitions:

Explanation of the state diagram: States are depicted as nodes. The initial state is I, the final state is F. A state transition is depicted as a directed arrow with the initiating event as a label, e. g. from I to S1 with event a, and is also written as a triple (I,a,S1). Note: A test case is a sequence of events that initiates the corresponding sequence of state transitions.

The state diagram contains the following 7 state transitions:

(I, a, S1)

(S1, a, S2), (S1, b, S3), (S1, c, F)

(S2, a, S3)

(S3, a, S2), (S3, b, F)

What is the minimum number of test cases to achieve 100% coverage of all valid state transitions in the diagram?

A. 2

B. 3

C. 4

D. 8

ANSWER: B

Explanation:

3 is the minimum because transition coverage requires every valid transition to be exercised at least once by the complete test suite. One test case must take the sequence `a, c`, covering the path from I through S1 to F using the direct transition labelled `c`. Since this reaches the final state, that transition cannot be combined with any of the other outgoing transitions from S1 in the same test case.

Two additional test cases are necessary to cover the distinct choices from S1 labelled `a` and `b`. For example, `a, a, a, b` covers I→S1, S1→S2, S2→S3, and S3→F. The sequence `a, b, a, a, b` covers I→S1, S1→S3, S3→S2, S2→S3, and S3→F. Together with `a, c`, these paths exercise all seven listed valid transitions. This applies the ISTQB state-transition-testing principle of deriving tests from a state model and measuring coverage over its transitions. See the ISTQB CTFL certification page and the ISTQB glossary entry for state transition testing.

QUESTION NO: 41

A test manager decided to skip static testing since he believes bugs can be found easily by doing dynamic testing. Was this decision right or wrong?

A. The decision was wrong. Ensuring quality mandates that static testing is performed after performing the dynamic testing.

B. The decision was right. Static testing is usually redundant if a product is planned to go through a full-cycle of dynamic testing.

C. The decision was right. Most of the bugs are easier to identify during the dynamic testing.

D. The decision was wrong. Static testing can find defects early in the development process, reducing the overall cost of testing and development

ANSWER: D

Explanation:

The decision was wrong. Static testing can find defects early in the development process, reducing the overall cost of testing and development. Static testing examines work products, such as requirements, user stories, designs, and code, without executing the software. This makes it possible to identify defects at the point where they were introduced, including ambiguous or inconsistent requirements, omissions, interface issues, and violations of standards. Finding and removing such defects early prevents them from propagating into later development artifacts and executable software.

Early defect detection is especially valuable because the effort and cost to diagnose, correct, retest, and manage a defect generally increase when it is discovered later in the lifecycle. Reviews and other static test techniques also promote shared understanding and improve the quality of the work product before dynamic testing begins. Dynamic testing remains important because it executes the software and can reveal failures and behavior-related problems, but it does not replace the preventative and early-detection benefits of static testing. The ISTQB Foundation Level syllabus explicitly treats static testing as a distinct test activity with important value in early defect detection and quality improvement. See the ISTQB Certified Tester Foundation Level overview and the ISTQB Glossary entry for static testing.

QUESTION NO: 42

The fact that defects are usually not evenly distributed among the various modules that make up a software application, but rather their distribution tend to reflect the Pareto principle:

A. is a false myth

B. is expressed by the testing principle referred to as Tests wear out '

C. is expressed by the testing principle referred to as ‘Defects cluster together’

D. is expressed by the testing principle referred to as ' Bug prediction '

ANSWER: C

Explanation:

“is expressed by the testing principle referred to as ‘Defects cluster together’” is correct because the defect clustering principle recognizes that defects are commonly concentrated in a relatively small number of components, features, modules, or areas of a system. This reflects the general idea behind the Pareto principle: a small proportion of causes or locations may account for a large proportion of observed effects. In testing, historical defect information, risk assessments, complexity, change frequency, and known problem areas can help identify where such concentrations are likely to occur.

This principle supports effective allocation of test effort. Testers can use evidence of clustering to prioritize deeper analysis, additional test conditions, more rigorous test techniques, and greater regression coverage in the areas with the greatest likelihood or impact of defects. It does not mean that untested or apparently low-defect areas can be ignored; rather, it provides a practical basis for risk-based prioritization when exhaustive testing is not feasible. The ISTQB CTFL syllabus lists defect clustering among its fundamental testing principles. See the ISTQB Certified Tester Foundation Level certification page and the CTFL syllabus resources.

QUESTION NO: 43

Which of the following is true about Oracles?

A. Sometimes old version of a product can be used as an Oracle

B. Oracles help in reproducing the irreproducible bugs

C. Oracles are derived from the design

D. Oracles can be generated automatically using data generators

ANSWER: A

Explanation:

Sometimes old version of a product can be used as an Oracle is correct. In ISTQB terminology, a test oracle is a source used to determine the expected result for a test input or to judge whether the observed behavior is correct. An earlier version of the same product can serve as such a source when it is sufficiently trusted and its behavior is accepted as the intended baseline for the scenarios being compared. The tester executes corresponding tests against the earlier and newer versions and compares their outcomes. This approach is commonly called back-to-back testing. It is especially useful where a complete, independently specified expected result would be difficult or expensive to determine for every test case. The approach still relies on an explicit assumption that the earlier version is suitable as a reference for the behavior under test; therefore, the tester should consider whether known defects, changed requirements, or intended functional differences affect that assumption. ISTQB describes a test oracle as a source for determining expected results, and its glossary includes an existing system as a potential oracle source. See the ISTQB Glossary definition of test oracle and the ISTQB CTFL v4.0 syllabus.