ISTQB Foundation Level Quick Guide

Chapter 1 Fundamentals of Testing

1.1 What is Software Testing?

- Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not.
- Testing is an investigation that can provide stakeholders with information about the quality of the product or service being tested. Testing is an activity that helps in identifying defects, errors, or missing requirements.

1.1.1 Typical Objectives of Testing

There are several typical objectives of testing:

- To identify defects and errors in the software product or system being tested.
- To ensure that the software product or system meets the specified requirements and works as expected.
- To verify that the software product or system works correctly, efficiently, and reliably.
- To improve the quality of the software product or system.
- To increase the confidence of stakeholders in the software product or system.
- To reduce the risks associated with using the software product or system.

1.1.2 Testing and Debugging:

Testing and debugging are two different activities that complement each other.

- Testing involves executing the software product or system to find defects, while debugging involves identifying and fixing the defects found during testing.
- Testing focused on evaluating the software product or system, while debugging is focused on identifying and fixing the issues discovered during testing.

1.2 Why is Testing Necessary?

Testing is necessary for several reasons:

- To ensure that the software product or system meets the specified requirements and works as expected
- To identify and fix defects before the software product or system is released to users.
- To improve the quality of the software product or system.
- To reduce the risks associated with using the software product or system.
- To increase the confidence of stakeholders in the software product or system.

1.2.1 Testing's Contributions to Success

Testing contributes to the success of a software product or system in several ways:

- It helps to identify and fix defects and errors before the software product or system is released to users.
- It improves the quality of the software product or system, leading to increased user satisfaction.
- It reduces the risks associated with using the software product or system, leading to increased user confidence.
- It helps to ensure that the software product or system meets the specified requirements and works as expected.
- It helps to identify opportunities for improvement in the software development process.

1.2.2 Quality Assurance and Testing

- Quality assurance is the process of ensuring that the software product or system meets the specified quality standards.
- Testing is a part of quality assurance that involves evaluating the software product or system to find defects and errors.
- Quality assurance includes other activities such as process improvement, code reviews, and requirements management.

1.2.3 Errors, Defects, and Failures

- An error is a human action that produces an incorrect or unexpected result.
- A defect is a flaw or imperfection in the software product or system that can cause it to fail to meet its requirements.
- A failure is the inability of the software product or system to perform its intended function.

- For example, an error could be a developer coding a feature incorrectly, leading to a defect in the software product. If this defect is not identified and fixed during testing, it could result in a failure when the software product is used by the end-users.

1.2.4 Defects, Root Causes and Effects

- A defect has a root cause, which is the underlying reason why the defect occurred.
- The effect is the result of the defect.
- Identifying the root cause of a defect helps in preventing similar defects from occurring in the future.

- For example, if a software product crashes when a user enters invalid data, the root cause of the defect could be an inadequate data validation routine. The effect of the defect is the software product crashing, which results in user frustration and potentially lost revenue.

1.3 Seven Testing Principles

The seven testing principles are a set of fundamental concepts that guide the testing process. These principles include:

1. Testing shows the presence of defects
2. Exhaustive testing is impossible
3. Early testing saves time and money
4. Defect clustering suggests that a small number of modules contain most of the defects
5. The Pareto principle applies to software testing, meaning that 80% of the defects are found in 20% of the modules
6. Testing is context-dependent
7. Absence-of-errors fallacy should not be used to measure the success of testing

1.4 Test Process

The test process is a set of activities and tasks that are performed to ensure that the software meets the requirements and is free of defects.

The test process includes the following phases:

1. Planning and control
2. Analysis and design
3. Implementation and execution
4. Evaluating exit criteria and reporting
5. Test closure activities

1.4.1 Test Process in Context

The test process needs to be adapted to the specific context of the project. The context factors that influence the test process include the following:

Objectives of testing:

1. Software development lifecycle model
2. Risks associated with the software
3. Business and technical constraints
4. Regulatory and legal requirements

1.4.2 Test Activities and Tasks

Test activities are a set of actions and tasks that are performed during the testing process. Test activities can be divided into the following categories:

1. Planning and control
2. Analysis and design
3. Implementation and execution
4. Evaluating exit criteria and reporting
5. Test closure activities

-: Test tasks are specific activities that are performed during each test activity.

Test tasks can include:

1. Reviewing the test basis
2. Identifying test conditions
3. Creating test cases
4. Executing test cases
5. Reporting defects

1.4.3 Test Work Products

- Test work products are the deliverables that are created during the testing process.
- **Test work products can include:**
  1. Test plan
  2. Test design specification
  3. Test case specification
  4. Test procedure specification
  5. Test log
  6. Test summary report

1.4.4 Traceability between the Test Basis and Test Work Products

- Traceability is the ability to follow a requirement or other work product through the development process to its final implementation.
- Traceability between the test basis and test work products ensures that all requirements are covered by test cases, and that all test cases are traceable back to the requirements.
- This traceability helps to ensure that all requirements are tested and that the testing is complete.

1.5 The Psychology of Testing

- The psychology of testing is the study of the cognitive and emotional factors that influence the testing process.
- Understanding the psychology of testing can help testers to improve the effectiveness of their testing.

- **Factors that can influence the psychology of testing include:**
  1. Attention and focus
  2. Perception and interpretation
  3. Memory and recall
  4. Motivation and morale
  5. Bias and heuristics

1.5.1 Human Psychology and Testing

- Human psychology and testing are closely related because testing is a human activity that requires cognitive and emotional skills.
- Testers need to understand the human factors that can influence testing in order to improve the effectiveness of their testing.

1.5.2 Tester’s and Developer’s Mindsets

- Testers and developers have different mindsets that can influence their approach to testing.
- Testers tend to focus on finding defects, while developers tend to focus on implementing functionality.
- Testers need to understand the developer’s mindset in order to effectively communicate defects and ensure that they are fixed.
- Developers need to understand the tester’s mindset in order to effectively prioritize and fix defects.

Chapter 2 Testing Throughout the Software Development Lifecycle

2.1 Software Development Lifecycle Models:

- Software Development Lifecycle Models describe the phases of the software development process, and each phase may require different testing types.
- Examples of software development lifecycle models include Waterfall model, V-model, Agile model, and Spiral model.
- Each of these models has its own strengths and weaknesses, and organizations may choose a model based on their specific needs and circumstances.

2.1.1 Software Development and Software Testing:

- Software development is the process of creating software from scratch.
- Software testing is the process of verifying and validating that software is working correctly, meets user requirements, and is free from defects.

2.1.2 Software Development Lifecycle Models in Context:

- Software Development Lifecycle Models provides a context for testing.
- Testing should be carried out throughout the development process to ensure that the software is delivered according to user requirements and is of high quality.
- The testing activities should be aligned with the software development lifecycle model.

2.2 Test Levels:

- Test Levels are the stages at which the testing is performed.
- Each level has specific objectives and test types.

- **The four test levels are:**
  1. Component Testing,
  2. Integration Testing,
  3. System Testing,
  4. Acceptance Testing.

2.2.1 Component Testing:

- Component Testing is also known as Unit Testing or Module Testing.
- It is the testing of individual software components, such as functions or procedures, in isolation from the rest of the system.

2.2.2 Integration Testing:

- Integration Testing is the process of combining the individual components and testing the interactions between them.
- It ensures that the software components work together as expected and that any defects or issues resulting from their interactions are identified and resolved.

2.2.3 System Testing:

- System Testing is the process of testing the entire system as a whole.
- It verifies that the system meets the specified requirements and performs as expected.
- System Testing may include functional testing, non-functional testing, and other testing types.

2.2.4 Acceptance Testing:

- Acceptance Testing is the process of verifying that the software meets the user requirements and is acceptable for delivery.
- It is usually the final stage of testing and may include functional and non-functional testing types.

2.3 Test Types:

Test Types are the techniques used to verify and validate the software.

Different types of testing are required at different test levels to ensure complete testing of the software.

Examples of test types include:

1. Functional Testing,
2. Non-Functional Testing,
3. White-box Testing,
4. Change-related Testing, and others.

2.3.1 Functional Testing:

Functional Testing is the process of verifying that the software meets the functional requirements.

It is usually performed at the System and Acceptance testing levels.

2.3.2 Non-Functional Testing:

Non-Functional Testing is the process of verifying that the software meets the non-functional requirements, such as:

- performance,
- security,
- usability

-: It is usually performed at the System and Acceptance testing levels.

2.3.3 White-box Testing:

White-box testing, also known as structural testing, is a testing approach where the tester has access to the internal workings of the system being tested, including its code, data structures, and algorithms.

- The goal of white-box testing is to ensure that all the code paths are tested, and all possible outcomes and results are checked.

-: This type of testing is usually performed by developers or testers who have a programming background.

Examples of white-box testing techniques include:

- statement coverage,
- branch coverage,
- path coverage.

2.3.4 Change-related Testing:

Change-related testing is performed when changes are made to the software.

The goal of change-related testing is to ensure that the changes have not introduced any new defects or issues, and that the existing functionality has not been affected.

Change-related testing can be performed at different levels, such as:

- component,
- integration,
- system testing

can use different types of testing, such as:

- regression testing,
- functional testing,
- non-functional testing.

2.3.5 Test Types and Test Levels:

Test types and test levels are related to each other, as different test types are usually performed at different test levels.

For example, functional testing is usually performed at the system and acceptance test levels, while non-functional testing, such as performance testing and security testing, is usually performed at the system and integration test levels.

White-box testing is usually performed at the component and integration test levels.

It is important to choose the appropriate test types and test levels based on the project's objectives, requirements, and risks.

2.4 Maintenance Testing:

Maintenance testing is a type of testing that is performed after the software has been released and is in use by the end-users.

The purpose of maintenance testing is to identify and fix defects and issues that arise during the software's maintenance phase.

Maintenance testing can be triggered by different factors, such as user feedback, changes in the software environment, and the introduction of new hardware or software components.

Maintenance testing can use different techniques, such as regression testing, corrective testing, and preventive testing.

2.4.1 Triggers for Maintenance:

Maintenance testing can be triggered by various factors, such as:

- user feedback,
- changes in the software environment,
- the introduction of new hardware or software components.

-: User feedback can include bug reports, feature requests, and usability issues.

-: Changes in the software environment can include changes in the operating system, web browser, or database management system.

-: The introduction of new hardware or software components can include the addition of new peripherals or the integration of third-party libraries.

2.4.2 Impact Analysis for Maintenance:

-: Impact analysis for maintenance is the process of evaluating the impact of a change on the software and its components.

-: The purpose of impact analysis is to determine the scope of the change and identify the areas of the software that may be affected.

Impact analysis can use different techniques, such as:

- static analysis,
- dynamic analysis,
- traceability analysis.

Static analysis involves examining the source code and other software artifacts without executing the software.

Dynamic analysis involves executing the software with test cases and analyzing its behavior.

Traceability analysis involves tracing the requirements, design, and test artifacts to identify the impact of a change.

Chapter 3 Static Testing

Static testing is a technique that examines a software product or related documentation without executing the code.

This technique can be applied throughout the software development lifecycle and has several benefits, including the early detection of defects and the improvement of software quality.

3.1 Static Testing Basics:

-: Static testing is a testing technique that verifies a software product's defects or issues without running the software.

-: It focuses on the product's static attributes such as code, requirements, design documents, and specifications.

-: Static testing involves reviewing and analyzing documents, codes, or other work products in a structured manner.

3.1.1 Work Products that Can Be Examined by Static Testing:

There are several work products that can be examined using static testing, such as:

- requirements documents,
- design documents,
- test plans,
- test cases,
- code,
- user manuals, and other related documents.

3.1.2 Benefits of Static Testing:

Static testing has several benefits, including early detection of defects, improved software quality, cost-effectiveness, and increased efficiency.

It also helps in identifying defects that may be difficult to find during dynamic testing, such as boundary value analysis and equivalence partitioning.

3.1.3 Differences between Static and Dynamic Testing:

Static testing differs from dynamic testing in terms of its approach and objectives.

Dynamic testing is focused on the execution of code and the identification of defects during runtime, while static testing is focused on the verification of a software product's static attributes without executing the code.

3.2 Review Process:

The review process involves a group of individuals reviewing a software product or documentation to identify defects and improve quality.

This process can be formal or informal and can take place at different stages of the software development lifecycle.

3.2.1 Work Product Review Process:

The work product review process involves the following steps:

1. Planning the review
2. Kick-off meeting
3. Reviewing the work product
4. Issue identification and documentation
5. Rework and follow-up

3.2.2 Roles and Responsibilities in a Formal Review:

There are different roles involved in a formal review process, such as:

- the moderator,
- author,
- reviewer,
- scribe.

Each role has specific responsibilities, such as the moderator who ensures that the review process is conducted in a structured manner and the author who presents the work product being reviewed.

3.2.3 Review Types:

There are different types of reviews, including:

- informal reviews,
- formal reviews,
- technical reviews,
- walkthroughs.

Each type of review has its objectives and is conducted at different stages of the software development lifecycle.

3.2.4 Applying Review Techniques:

Review techniques include:

- walkthroughs,
- inspections,
- peer reviews.

These techniques can be applied to different types of work products and have different objectives, such as identifying defects or improving software quality.

3.2.5 Success Factors for Reviews:

Success factors for reviews include having clear objectives, a structured review process, adequate preparation, and ensuring that the right people are involved.

Other success factors include having a positive attitude, constructive feedback, and a willingness to learn and improve.

Chapter 4 Test Techniques

4.1 Categories of Test Techniques:

Test techniques can be broadly divided into three categories:

1. black-box techniques,
2. white-box techniques,
3. experience-based techniques.

Testers choose appropriate test techniques based on the software being tested, the requirements, and other factors such as time and resources.

4.1.1 Choosing Test Techniques:

The decision of which test technique to use depends on several factors, including

1. system's complexity,
2. risks associated with the system,
3. availability of documentation,
4. testing objectives,
5. skill level of the testing team.

Testers may use a combination of different techniques to achieve better test coverage.

4.1.2 Categories of Test Techniques and Their Characteristics:

Black-box techniques focus on testing the system without knowledge of its internal workings, while white-box techniques involve testing the system with knowledge of its internal workings.

Experience-based techniques involve testing based on the tester's experience, intuition, and creativity.

4.2 Black-box Test Techniques:

Black-box test techniques are designed to test the system without knowledge of its internal workings.

These techniques are useful in testing the functionality of the system and ensuring that it meets the requirements.

Examples of black-box test techniques include equivalence partitioning, boundary value analysis, decision table testing, state transition testing, and use case testing.

4.2.1 Equivalence Partitioning:

Equivalence partitioning is a black-box test technique that involves dividing the input domain of a system into groups of equivalent inputs.

The idea is to select a representative test case from each group, which should provide similar testing results.

For example, if a system accepts inputs in the range of 1 to 100, then the input domain can be divided into three equivalence classes: inputs less than 1, inputs between 1 and 100, and inputs greater than 100.

4.2.2 Boundary Value Analysis:

Boundary value analysis is a black-box test technique that involves testing the boundaries of an input domain.

The idea is to test the values at the boundaries of the input domain, as these are likely to be more prone to errors.

For example, if a system accepts inputs in the range of 1 to 100, then the boundary values to be tested would be 1, 100, and values just below or above these limits.

4.2.3 Decision Table Testing:

Decision table testing is a black-box test technique that involves creating a table to represent the combinations of inputs and expected outputs for a given set of business rules.

The tester can then use the decision table to derive test cases that cover all the possible combinations of inputs and expected outputs.

4.2.4 State Transition Testing:

State transition testing is a black-box test technique that involves testing the behavior of a system as it moves from one state to another.

The tester identifies the various states of the system and the events that cause it to transition from one state to another.

The tester then creates test cases to ensure that the system behaves correctly during these transitions.

4.2.5 Use Case Testing:

Use case testing is a black-box test technique that involves testing the system's behavior from the user's perspective.

The tester identifies the various use cases for the system and creates test cases to ensure that the system behaves correctly for each use case.

4.3 White-box Test Techniques:

White-box test techniques, also known as structural or code-based techniques, focus on the internal workings of the software being tested.

These techniques are primarily used for testing software at the unit, integration, and system levels, and rely on the knowledge of the internal code and structure of the software.

Some of the commonly used white-box techniques are statement coverage, decision coverage, and condition coverage.

4.3.1 Statement Testing and Coverage:

Statement coverage is a white-box technique that ensures each statement in the code is executed at least once during testing.

It involves creating test cases that execute each statement in the code, ensuring that all statements are covered.

Statement coverage is a useful technique for detecting issues with the flow of control in the code.

4.3.2 Decision Testing and Coverage:

Decision coverage is a white-box technique that ensures that all possible decisions in the code have been tested.

A decision is a point in the code where the program has a choice of two or more possible paths.

Decision coverage involves creating test cases that ensure that all decisions in the code have been exercised, both true and false.

4.3.3 The Value of Statement and Decision Testing:

Statement and decision testing are important techniques for verifying the correctness of the code.

These techniques can help to identify issues with the flow of control in the code, and ensure that all possible paths through the code have been tested.

4.4 Experience-based Test Techniques:

Experience-based test techniques rely on the knowledge, skill, and intuition of the testers to identify potential issues with the software being tested.

These techniques are typically used in situations where the requirements are unclear or incomplete, or where the software is complex or difficult to test.

4.4.1 Error Guessing:

Error guessing is an experience-based test technique that involves using the tester's knowledge and experience to guess what types of errors might occur in the software being tested.

The tester then creates test cases that specifically target those types of errors. This technique can be particularly effective in identifying issues that may not be apparent from the requirements or design documentation.

4.4.2 Exploratory Testing:

Exploratory testing is an experience-based test technique that involves ad-hoc testing of the software being tested, based on the tester's intuition and knowledge.

This technique is particularly useful in situations where the software is complex or difficult to test, or where the requirements are unclear or incomplete.

4.4.3 Checklist-based Testing:

Checklist-based testing is an experience-based test technique that involves using a predefined checklist of items to be tested.

The checklist can include items such as common errors, performance issues, or usability issues.

The tester then creates test cases that specifically target the items on the checklist.

This technique can be particularly useful in ensuring that important areas of the software are not overlooked during testing.

Chapter 5 Test Management

5.1 Test Organization:

Independent testing refers to testing conducted by a group that is separate from the development team.

This separation can help to increase objectivity, improve quality, and reduce the possibility of conflicts of interest.

Independent testers may be part of a separate team or organization, or they may work within the same organization but report to a different manager.

The tasks of a test manager may include defining the test strategy, planning and controlling the testing effort, allocating resources, and communicating with stakeholders.

The tasks of a tester may include designing test cases, executing tests, reporting defects, and analyzing test results.

5.2 Test Planning and Estimation:

A test plan is a document that describes the objectives, scope, approach, and schedule of the testing effort.

It should include details on the test strategy, test approach, entry criteria, exit criteria, and test execution schedule.

The test plan should be reviewed and approved by all relevant stakeholders.

Test estimation involves predicting the amount of time, effort, and resources required to complete the testing effort.

This can be challenging due to the uncertainty involved in software development and the complexity of testing.

Test estimation techniques include

1. expert judgment,
2. historical data,
3. metrics-based approaches.

5.3 Test Monitoring and Control:

Test monitoring involves collecting and analyzing data on the testing effort to determine its status and progress.

Test control involves taking corrective action to address issues that arise during testing.

Test metrics can be used to monitor and control the testing effort, such as defect metrics, test coverage metrics, and test progress metrics.

Test reports provide information on the testing effort to stakeholders, including test results, defects found, and progress made.

The contents of a test report may vary depending on the audience, but should be accurate, relevant, and timely.

5.4 Configuration Management:

Configuration management is the process of identifying, organizing, and controlling changes to the software being tested, including its documentation, requirements, design, code, and test artifacts.

Testers must ensure that changes to the software do not affect the testing effort and that any changes made during the testing process are controlled and traceable.

Configuration management tools and processes are used to manage and maintain the various software versions and artifacts, and to ensure that only authorized individuals make changes to them.

5.5 Risks and Testing:

In the context of testing, risk refers to the probability of a failure occurring and the impact of that failure on the software, its users, and the organization as a whole.

Risk-based testing involves identifying the risks that could affect the software and using that information to determine the types of tests that should be performed.

Testers must understand the different types of risks and how they can impact testing, and they must work with the project team to develop a risk management plan.

This plan should include :

1. risk identification,
2. analysis,
3. mitigation strategies.

5.6 Defect Management:

Defect management involves the process of identifying, documenting, tracking, and resolving defects found during the testing process.

Testers must use a defect management tool to log defects, track their status, and communicate with the development team about their resolution.

Defects should be categorized based on their severity, and priorities should be assigned based on the impact they have on the software.

Testers must also perform root cause analysis to determine the underlying cause of the defect and to prevent similar defects from occurring in the future.

Defect metrics can be used to track and report on the effectiveness of the defect management process.

Chapter 6 Tool Support for Testing

6.1.1 Test Tool Classification:

Test tools can be broadly classified into four categories:

1. Test management tools: these tools help in planning, organizing, and managing the testing activities.
2. Static testing tools: these tools help in performing static testing techniques like reviews and inspections.
3. Test design tools: these tools help in designing test cases and test scenarios.
4. Test execution tools: these tools help in executing the test cases, capturing the results, and generating test reports.

6.1.2 Benefits and Risks of Test Automation:

The benefits of test automation include:

1. Increased test coverage: automated tests can cover more scenarios than manual testing.
2. Improved test accuracy: automated tests are less prone to errors than manual testing.
3. Reusability: automated tests can be reused multiple times.
4. Cost reduction: automated tests can save costs in the long run.

The risks of test automation include:

1. High initial cost: automation tools can be expensive, and setting up the automation environment can be time-consuming.
2. Maintenance overhead: automated tests require maintenance, and any changes in the application can break the automated tests.
3. Limited scope: automation cannot replace manual testing completely, as certain types of testing like exploratory testing require human intuition.

6.2.1 Main Principles for Tool Selection:

The main principles for selecting a tool are:

1. Identify the problem to be solved: the tool should be selected based on the specific problem it solves.
2. Evaluate the tool against the requirements: the tool should meet the functional and non-functional requirements.
3. Ensure compatibility with the existing environment: the tool should be compatible with the existing tools and technologies.
4. Assess the cost and benefits: the tool should provide significant benefits for the cost incurred.

6.2.2 Pilot Projects for Introducing a Tool into an Organization:

- Pilot projects are useful to introduce a new tool into an organization.
- The pilot project involves selecting a small project or a subset of the larger project and using the tool to test it.
- The pilot project provides the opportunity to evaluate the tool's effectiveness, identify any issues, and make changes as necessary.

6.2.3 Success Factors for Tools:

The success of a tool depends on several factors:

- Tool selection: selecting the right tool for the right problem.
- Tool customization: customizing the tool to fit the specific needs of the organization.
- Training: providing adequate training to the users to effectively use the tool.
- Tool support: ensuring that the tool is supported and maintained properly.
- Tool integration: integrating the tool with other tools and technologies used in the organization.

Question #1 (1 Point)

Which one of the following is the BEST description of a test condition?

a) An attribute of a component or system specified or implied by requirements documentation.
b) An aspect of the test basis that is relevant to achieve specific test objectives.
c) The capability of the software product to provide functions which meet stated and implied needs when the software is used under specified conditions.
d) The percentage of all single condition outcomes that independently affect a decision outcome that have been exercised by a test case suite.

Select one option.

FL-1.x (K1) Keywords Chapter 1

Justification

a) Not correct – Definition of feature according to Glossary.
b) **Correct** – From Glossary.
c) Not correct – Definition of functionality according to Glossary.
d) Not correct – Definition of modified condition decision coverage according to Glossary.

Question #2 (1 Point)

Which of the following statements is a valid objective for testing?

a) To determine whether enough component tests were executed within system testing.
b) To find as many failures as possible so that defects can be identified and corrected.
c) To prove that all possible defects are identified.
d) To prove that any remaining defects will not cause any failures.

Select one option.

FL-1.1.1 (K1) Identify typical objectives of testing

Justification

a) Not correct – Component testing is not part of System testing.
b) **Correct** – Syllabus 1.1.1
c) Not correct – Principle #1 states that exhaustive testing is impossible, so one can never prove that all defects were identified.
d) Not correct – To make an assessment whether a defect will cause a failure or not, one has to detect the defect first. Saying that no remaining defect will cause a failure, implicitly means that all defects were found. This contradicts Principle #1.

Question #3 (1 Point)

Which of the following statements correctly describes the difference between testing and debugging?

a) Testing identifies the source of defects; debugging analyzes the defects and proposes prevention activities.
b) Testing shows failures caused by defects; debugging finds, analyzes, and removes the causes of failures in the software.
c) Testing removes faults; debugging identifies the causes of failures.
d) Testing prevents the causes of failures; debugging removes the failures.

Select one option.

FL-1.1.2 (K2) Differentiate testing from debugging

Justification

a) Not correct. Testing does not identify the source of defects.
b) **Correct**. Syllabus 1.1.2: Executing tests can show failures that are caused by defects in the 8 software. Debugging is the development activity that finds, analyzes, and fixes such defects.
c) Not correct. Testing does not remove faults.
d) Not correct. Testing does not directly prevent the causes of failures. Debugging does not removes the failures, only the causes of failures

Question #4 (1 Point)

Which one of the statements below describes a failure discovered during testing or in production?

a) The product crashed when the user selected an option in a dialog box.
b) The wrong version of one source code file was included in the build.
c) The computation algorithm used the wrong input variables.
d) The developer misinterpreted the requirement for the algorithm.

Select one option.

FL-1.2.3 (K2) Distinguish between error, defect and failure

Justification

a) **Correct** – A failure is an external manifestation of a defect. A crash is clearly noticeable by the user.
b) Not correct – This is a defect, not a failure, since there is something wrong in the code. It may not result in a failure, for example if the changes in the source code file are only in comments.
c) Not correct – This is a defect, not a failure, as there is a flaw in the code implementing the algorithm. If this computation is not used in a test or in production, a failure will not occur.
d) Not correct – This is an error, not a failure. The misinterpretation of the requirement may or may not lead to a defect in the impementation of the algorithm, which in turn may or may not lead to a failure.

Question #5 (1 Point)

Which of the following statements CORRECTLY describes one of the seven key principles of software testing?

a) By using automated testing it is possible to test everything.
b) With sufficient effort and tool support, exhaustive testing is feasible for all software.
c) It is impossible to test all input and precondition combinations in a system.
d) The purpose of testing is to prove the absence of defects.

Select one option.

FL-1.3.1 (K2) Explain the seven testing principles

Justification

a) Not Correct – Exhaustive testing is impossible, regardless of it being manual or automated.
b) Not Correct– Exhaustive testing is impossible, regardless of the amount of effort put into testing.
c) **Correct** – Syllabus 1.3: Principle #2 says “Testing everything (all combinations of inputs and preconditions) is not feasible except for trivial cases”.
d) Not Correct– This statement is contradicting Principle #1 says “Testing shows the presence of defects: Testing can show that defects are present, but cannot prove that there are no defects”.

Question #6 (1 Point)

In what way can testing be part of Quality assurance?

a) It ensures that requirements are detailed enough.
b) It reduces the level of risk to the quality of the system.
c) It ensures that standards in the organization are followed.
d) It measures the quality of software in terms of number of executed test cases.

Select one option.

FL-1.2.2 (K2) Describe the relationship between testing and quality assurance and give examples of how testing contributes to higher quality

Justification

a) Not correct – This is Quality assurance but not testing.
b) **Correct** – Syllabus 1.2.2. Testing contributes to the achievement of quality in a variety of ways.
c) Not correct – This is Quality assurance but not testing.
d) Not correct – The quality can not be measured by counting the number of executed test cases without knowing the outcome.

Question #7 (1 Point)

Which of the below tasks is performed during the test analysis activity of the test process?

a) Identifying any required infrastructure and tools.
b) Creating test suites from test scripts.
c) Analyzing lessons learned for process improvement.
d) Evaluating the test basis for testability.

Select one option.

FL-1.4.2 (K2) Describe the test activities and respective tasks within the test process

Justification

a) Not correct – this activity is performed during the Test design activity.
b) Not correct – this activity is performed during the Test implementation activity.
c) Not correct – this activity is performed during the Test completion activity.
d) **Correct** – this activity is performed during the Test analysis activity. Syllabus 1.4.2.

Question #8 (1 Point)

Differentiate the following test work products,1-4, by mapping them to the right description, A-D.

1. Test suite.
2. Test case.
3. Test script.
4. Test charter.

A. A group of test scripts or test execution schedule.
B. A set of instructions for the automated execution of test procedures.
C. Contains expected results.
D. An event that could be verified.

a) 1A, 2C, 3B, 4D.
b) 1D, 2B, 3A, 4C.
c) 1A, 2C, 3D, 4B.
d) 1D, 2C, 3B, 4A.

Select one option.

FL-1.4.3 (K2) Differentiate the work products that support the test process

Justification

Test suite: Syllabus 1.4.3 for Test implementation:
Test implementation work products also include test suites, which are groups of test scripts, as well as a test execution schedule. (1A).
Test case: Glossary, A set of input values, execution preconditions, **expected results** and execution postconditions…. (2C).
Test script: Glossary test script, A set of instructions for the automated execution of test procedures (3B).
Test charter: Glossary, A statement of test objectives, and possibly test ideas about how to test. Test charters are used in exploratory testing. (4D).

Thus:

a) **Correct**
b) Not correct
c) Not correct
d) Not correct

Question #9 (1 Point)

How can white-box testing be applied during acceptance testing?

a) To check if large volumes of data can be transferred between integrated systems.
b) To check if all code statements and code decision paths have been executed.
c) To check if all work process flows have been covered.
d) To cover all web page navigations.

Select one option.

FL-2.3.2 (K1) Recognize that functional, non-functional and white-box tests occur at any test level

Justification

a) Not correct – Relevant for integration testing.
b) Not correct – Relevant for component testing.
c) **Correct** – Syllabus 2.3.5: For acceptance testing, tests are designed to cover all supported financial data file structures and value ranges for bank-to-bank transfers.
d) Not correct – Relevant for system testing.

Question #10 (1 Point)

Which of the following statements comparing component testing and system testing is TRUE?

a) Component testing verifies the functionality of software modules, program objects, and classes that are separately testable, whereas system testing verifies interfaces between components and interactions between different parts of the system.
b) Test cases for component testing are usually derived from component specifications, design specifications, or data models, whereas test cases for system testing are usually derived from requirement specifications, or use cases.
c) Component testing only focuses on functional characteristics, whereas system testing focuses on functional and non-functional characteristics.
d) Component testing is the responsibility of the testers, whereas system testing typically is the responsibility of the users of the system.

Select one option.

FL-2.2.1 (K2) Compare the different test levels from the perspective of objectives, test basis, test objects, typical defects and failures, and approaches and responsibilities

Justification

a) Not correct – System testing does not test interfaces between components and interactions between different parts of the system; this is a target of integration tests.
b) **Correct** – Syllabus 2.2.1: Examples of work products that can be used as a test basis for component testing include: detailed design, code, data model, component specifications.
Syllabus 2.2.3: Examples of work products for system testing include: System and software requirement specifications (functional and non-functional), …,.. use cases.
c) Not correct – Component testing does not ONLY focus on functional characteristics.
d) Not correct – Component testing typically is the responsibility of the developers, whereas system testing typically is the responsibility of testers.

Question #11 (1 Point)

Which one of the following is TRUE?

a) The purpose of regression testing is to check if the correction has been successfully implemented, while the purpose of confirmation testing is to confirm that the correction has no side effects.
b) The purpose of regression testing is to detect unintended side effects, while the purpose of confirmation testing is to check if the system is still working in a new environment.
c) The purpose of regression testing is to detect unintended side effects, while the purpose of confirmation testing is to check if the original defect has been fixed.
d) The purpose of regression testing is to check if the new functionality is working, while the purpose of confirmation testing is to check if the originally defect has been fixed.

Select one option.

FL-2.3.3 (K2) Compare the purposes of confirmation testing and regression testing

Justification

a) Not correct – Confirmation testing does not check successful implementation and confirmation testing does not check for side effects.
b) Not correct– The statement about confirmation testing should be about regression testing.
c) **Correct** – Syllabus 2.3.4
d) Not correct – Testing new functionality is not regression testing

Question #12 (1 Point)

Which one of the following is the BEST definition of an incremental development model?

a) Defining requirements, designing software and testing are done in a series with added pieces.
b) A phase in the development process should begin when the previous phase is complete.
c) Testing is viewed as a separate phase which takes place after development has been completed.
d) Testing is added to development as an increment.

Select one option.

FL-2.1.1 (K2) Explain the relationship between software development activities and test activities in the software life cycle

Justification

a) **Correct**– Syllabus 2.1.1: Incremental development involves establishing requirements, designing, building, and testing a system in pieces.
b) Not correct – This is a sequential model.
c) Not correct – This describes the Waterfall model.
d) Not correct – Testing alone is not an increment in the development.

Question #13 (1 Point)

Which of the following should **NOT** be a trigger for maintenance testing?

a) Decision to test the maintainability of the software.
b) Decision to test the system after migration to a new operating platform.
c) Decision to test if archived data is possible to be retrieved.
d) Decision to test after "hot fixes".

Select one option.

FL-2.4.1 (K2) Summarize triggers for maintenance testing

Justification

a) **Correct** – this is maintainability testing, not maintenance testing.
b) Not correct – this is a trigger for maintenance testing, see the syllabus chapter 2.4.1: Operational tests of the new environment as well as of the changed software.
c) Not correct – this a the trigger for maintenance testing, see the syllabus chapter 2.4.1: Testing restore/retrieve procedures after archiving for long retention periods.
d) Not correct – this a the trigger for maintenance testing, see the syllabus chapter 2.4.1: Reactive modification of a delivered software product to correct emergency defects that have caused actual failures.

Question #14 (1 Point)

Which of the following options are roles in a formal review?

a) Developer, Moderator, Review leader, Reviewer, Tester.
b) Author, Moderator, Manager, Reviewer, Developer.
c) Author, Manager, Review leader, Reviewer, Designer.
d) Author, Moderator, Review leader, Reviewer, Scribe.

Select one option.

FL-3.2.2 (K1) Recognize the different roles and responsibilities in a formal review

Justification

a) Not correct – Tester and developer are NOT roles as per Syllabus, section 3.2.2.
b) Not correct – Developer is NOT a role as per Syllabus, section 3.2.2.
c) Not correct – Designer is NOT a role as per Syllabus, section 3.2.2.
d) **Correct** –see Syllabus, section 3.2.2.

Question #15 (1 Point)

Which of the following describes the main activities of a formal review?

a) Initiation, backtracking, individual review, issue communication and analysis rework, follow-up.
b) Planning, individual review, issue communication and analysis, rework, closure, follow-up.
c) Planning, initiate review, individual review, issue communication and analysis, fixing and reporting.
d) Individual review, issue communication and analysis, rework, closure, follow-up, root cause analysis.

Select one option.

FL-3.2.1 (K2) Summarize the activities of the work product review process

Justification

a) Not correct – See c) for the activities in the review process.
b) Not correct – See c) for the activities in the review process.
c) **Correct** – According to Syllabus chapter 3.2.1: planning, initiate review, individual review, issue communication and analysis, fixing defects and report.
d) Not correct – See c) for the activities in the review process.

Question #16 (1 Point)

Which of the review types below is the BEST option to choose when the review must follow a formal process based on rules and checklists?

a) Informal Review.
b) Technical Review.
c) Inspection.
d) Walkthrough.

Select one option.

FL-3.2.3 (K2) Explain the differences between different review types: informal review, walkthrough, technical review and inspection

Justification

a) Not correct – Informal review does not use a formal process.
b) Not correct – Use of checklists are optional.
c) **Correct** – As per Syllabus 3.2.3: Formal process based on rules and checklists.
d) Not correct – Does not explicitly require a formal process.

Question #17 (1 Point)

Which TWO of the following statements about static testing are MOST true?

a) A cheap way to detect and remove defects.
b) It makes dynamic testing less challenging.
c) Early validation of user requirements.
d) It makes it possible to find run-time problems early in the lifecycle.
e) When testing safety-critical system, static testing has less value because dynamic testing finds the defects better.

Select two options.

FL-3.1.2 (K2) Use examples to describe the value of static testing

Justification

a) Correct – Syllabus 3.1.2: Defects found early are often much cheaper to remove than defects detected later in the lifecycle.
b) Not correct – Dynamic testing still has its challenging objectives
c) **Correct** – Syllabus 3.1.2: Preventing defects in design or coding by uncovering omissions, inaccuracies, inconsistencies, ambiguities, and redundancies in requirements.
d) Not correct – This is dynamic testing.
e) Not correct – Static analysis is important for safety-critical computer systems. Syllabus 3.1.

Question #18 (1 Point)

The design of a newspaper subscriptions system is being reviewed. The expected system users are:

- Subscribers
- Technical support team
- Billing department
- Database administrator

Each type of user logs into the system through a different login interface (e.g. subscribers login via a web page; technical support via an application).

Different reviewers were requested to review the system's login flow from the perspective of the above user categories.

Which of the following review comments is MOST LIKELY to have been made by all reviewers?

a) The login page on the web is cluttered with too much advertisement space. As a result, it is hard to find the "forgot password?" link.
b) The login to access the billing information should also allow access to subscribers' information and not force a second login session.
c) After logging-in to the database application, there is no log-out function.
d) The log in flow is un-intuitive since it requires entering the password first, before the user name can be keyed-in.

Select one option.

FL-3.2.4 (K3) Apply a review technique to a work product to find defects

Justification

a) Not correct – this impacts only the subscribers; possibly others but for sure not the technical support since they don't access the data via a web page.
b) Not correct – this comment would come from the review that took the perspective of the billing department, but not from other reviewers.
c) Not correct – this comment would come from the review that took the perspective of the database administrator, but not from other reviewers.
d) **Correct** – Every type of user must be authenticated before accessing to the system, so all users of the system would note (and suffer) an un-intuitive login flow.

Question #19 (1 Point)

What is checklist-based testing?

a) A test technique in which tests are derived based on the tester's knowledge of past failures, or general knowledge of failure modes.
b) Procedure to derive and/or select test cases based on an analysis of the specification, either functional or non-functional, of a component or system without reference to its internal structure.
c) An experience-based test technique whereby the experienced tester uses a high-level list of items to be noted, checked, or remembered, or a set of rules or criteria against which a product has to be verified.
d) An approach to testing where the tester dynamically designs and executes tests based on their knowledge, exploration of the test item and the results of previous tests.

Select one option.

FL-4.x (K1) Keywords

Justification

a) Not correct – This is error giessing,defined in Glossary.
b) Not correct – This is black-box test technique, defined in Glossary.
c) **Correct** – Defined in Glossary.
d) Not correct – This is exploratory testing, defined in Glossary.

Question #20 (1 Point)

Which one of the following options is categorized as a black-box test technique?

a) Techniques based on analysis of the architecture.
b) Techniques checking that the test object is working according to the technical design.
c) Techniques based on the expected use of the software.
d) Techniques based on formal requirements.

Select one option.

FL-4.1.1 (K2) Explain the characteristics, commonalities, and differences between black-box test techniques, white-box test techniques and experience-based test techniques

Justification

a) Not correct – This is a white-box test technique.
b) Not correct – This is a white-box test technique.
c) Not correct – This is a experience-based test technique.
d) **Correct** – Syllabus 4.1.2: Black-box test techniques (also called behavioral or behavior-based techniques) are based on an analysis of the appropriate test basis (e.g. formal requirements documents, specifications, use cases, user stories).

Question #21 (1 Point)

The following statement refers to decision coverage:
"When the code contains only a single 'if' statement and no loops or CASE statements, any single test case we run will result in 50% decision coverage."
Which of the following sentences is correct?

a) The sentence is true. Any single test case provides 100% statement coverage and therefore 50% decision coverage.
b) The sentence is true. Any single test case would cause the outcome of the "if" statement to be either true or false.
c) The sentence is false. A single test case can only guarantee 25% decision coverage in this case.
d) The sentence is false. The statement is too broad. It may be correct or not, depending on the tested software.

Select one option.

FL-4.3.2 (K2) Explain decision coverage

a) Not correct – While the given statement is true, the explanation is not.
b) **Correct** – Since any test will cause the outcome of the "if" statement to be either TRUE or FALSE, by defintion we achieved 50% decision coverage.
c) Not correct – A single test can give more than 25% decision coverage.
d) Not correct – The statement is specific and always true.

Question #22 (1 Point)

Which one of the following is the BEST description of statement coverage?

a) It is a metric which is used to calculate and measure the percentage of test cases that have been executed.
b) It is a metric, which is used to calculate and measure the percentage of statements in the source code which have been executed.
c) It is a metric, which is used to calculate and measure the number of statements in the source code which have been executed by test cases that are passed.
d) It is a metric that give a true/false confirmation if all statements are covered or not.

Select one option.

FL-4.3.1 (K2) Explain statement coverage

Justification

a) Not correct – Statement coverage measures the percentage of statements exercised by test cases.
b) **Correct** – Syllabus 4.3.1: Statement testing exercises the executable statements in the code. Coverage is measured as the number of statements executed by the tests divided by the total number of executable statements in the test object, normally expressed as a percentage.
c) Not correct – The coverage does not measure pass/fail.
d) Not correct – It is a metric, not a true/false.

Question #23 (1 Point)

Which TWO of the following statements about the relationship between statement coverage and decision coverage are true?

a) Decision coverage is stronger than statement coverage.
b) Statement coverage is stronger than decision coverage.
c) 100% statement coverage guarantees 100% decision coverage.
d) 100% decision coverage guarantees 100% statement coverage.
e) Decision coverage can never reach 100%.

Select two options.

FL-4.3.3 (K2) Explain the value of statement and decision coverage

Justification

See syllabus chapter 4.3.3: Achieving 100% decision coverage guarantees 100% statement coverage (but not vice versa).

Thus

a) **Correct –** The statement is true.
b) Not correct **–** The statement is false.
c) Not correct**–** The statement is false.
d) **Correct –** The statement is true.
e) Not correct **–** The statement is false.

Question #24 (1 Point)

Which of the following situations is NOT suited for using exploratory testing?

a) When there is time pressure, and/or the requirements are incomplete or inapplicable
b) When the system is developed and tested incrementally.
c) When only new and inexperienced testers are available.
d) When the main part of the application can be tested only at the customer's site.

Select one option.

FL-4.4.2 (K2) Explain exploratory testing

Justification

a) Not correct – Syllabus 4.4.2: Exploratory testing is most useful when there are few or significant time pressure on testing.
b) Not correct – exploratory testing can be used here.
c) **Correct** – exploratory tests should be performed by experienced testers with knowledge of similar applications and technologies. The tester needs constantly to make decisions during exploratory testing, e.g.what to test next.
d) Not correct – exploratory testing can be used at any location.

Question #25 (1 Point)

An employee's bonus is to be calculated. It cannot be negative, but it can be calculated down to zero. The bonus is based on the length of employment.

The categories are: less than or equal to 2 years, more than 2 years but less than 5 years, 5 or more years, but less than 10 years, 10 years or longer.

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

a) 3.
b) 5.
c) 2.
d) 4.

Select one option.

FL-4.2.1 (K3) Apply equivalence partitioning to derive test cases from given requirements

Justification

a) Not correct – see the correct partitions in d).
b) Not correct – see the correct partitions in d).
c) Not correct – see the correct partitions in d).
d) **Correct** – Partions as below:
    1. equivalence partition: $0 <$ employment time ≤ 2.
    2. equivalence partition: $2 <$ employment time < 5.
    3. equivalence partition: $5 \leq$ employment time < 10.
    4. equivalence partition: $10 \leq$ employment time.

Question #26 (1 Point)

A speed control and reporting system has the following characteristics:
If you drive 50 km/h or less, nothing will happen.
If you drive faster than 50 km/h, but 55 km/h or less, you will be warned.
If you drive faster than 55 km/h but not more than 60 km/h, you will be fined.
If you drive faster than 60 km/h, your driving license will be suspended.

Which would be the most likely set of values (km/h) identified by two-point boundary value analysis?

a) 0, 49, 50, 54, 59, 60.
b) 50, 55, 60.
c) 49, 50, 54, 55, 60, 62.
d) 50, 51, 55, 56, 60, 61.

Select one option.

FL-4.2.2 (K3) Apply boundary value analysis to derive test cases from given requirements

Justification

The following partitions can be identified:

1. – 50 Two-point boundaries 50, 51
2. 51 – 55 Two-point boundaries 50, 51, 55, 56
3. 56 – 60 Two-point boundaries 55, 56, 60, 61
4. 61 – Two-point boundaries 60, 61

Thus:

a) Not correct – Does not include all two-point boundary values. Also includes values not necessary for two-point boundary value analysis.
b) Not correct – Does not include all two-point boundary values.
c) Not correct – Does not include all two-point boundary values. Also includes values not necessary for two-point boundary value analysis.
d) **Correct** – Includes all two-point boundary values

Question #27 (1 Point)

A company's employees are paid bonuses if they work more than a year in the company and achieve individually agreed targets.

The following decision table has been designed to test the logic for paying bonuses:

		T1	T2	T3	T4	T5	T6	T7	T8
Conditions									
Cond1	Employment for more than 1 year?	YES	NO	YES	NO	YES	NO	YES	NO
Cond2	Agreed target?	NO	NO	YES	YES	NO	NO	YES	YES
Cond3	Achieved target?	NO	NO	NO	NO	YES	YES	YES	YES
Action									
	Bonus payment?	NO	NO	NO	NO	NO	NO	YES	NO

Which test cases could be eliminated in the above decision table because the test case wouldn't occur in a real situation?

a) T1 and T2.
b) T3 and T4.
c) T7 and T8.
d) T5 and T6.

Select one option.

FL-4.2.3 (K3) Apply decision table testing to derive test cases from given requirements

Justification

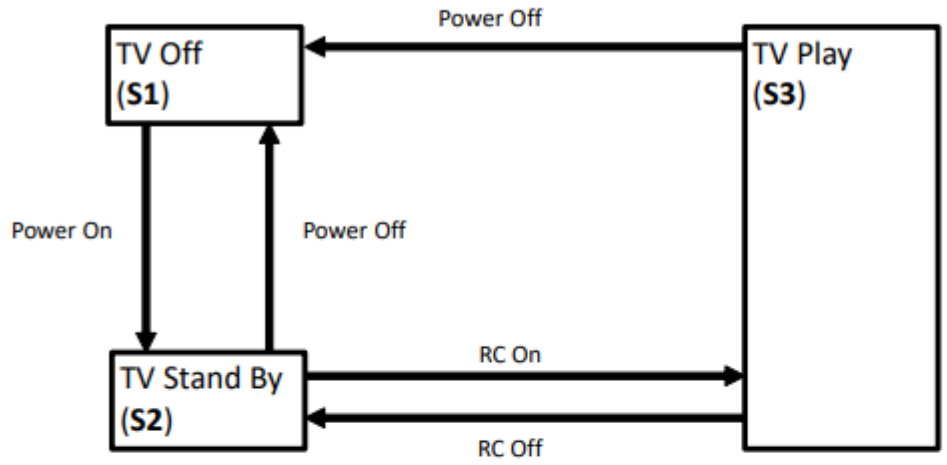
In the test cases T5 and T6 the situation described is logically impossible. If there was no agreement on targets, it's impossible to claim that the targets were reached. Since this situation can´t occur, we therefore can eliminate the corresponding test cases.

Hence

a) Not correct.
b) Not correct.
c) Not correct.
d) **Correct.**

Question #28 (1 Point)

Which of the following statements about the given state transition diagram and table of test cases is TRUE?



Test Case	1	2	3	4	5
Start State	S1	S2	S2	S3	S3
Input	Power On	Power Off	RC On	RC Off	Power Off
Expected Final State	S2	S1	S3	S2	S1

a) The given test cases can be used to cover both valid and invalid transitions in the state transition diagram.
b) The given test cases represent all possible valid transitions in the state transition diagram.
c) The given test cases represent only some of the valid transitions in the state transition diagram.
d) The given test cases represent sequential pairs of transitions in the state transition diagram.

Select one option.

FL-4.2.4 (K3) Apply state transition testing to derive test cases from given requirements

Justification

Proposed test case cover all five possible single valid transitions in the given state diagram

(S1->S2, S2->S1, S2->S3, S3->S2, S3->S1).

a) Not correct – because no invalid transitions are covered.
b) **Correct** – because all valid transitions are covered.
c) Not correct – because all valid transitions are covered.
d) Not correct – because the order in which the test cases are run has not been specified, we don't know what pairs of transitions will occur.

Question #29 (1 Point)

A video application has the following requirement:
The application shall allow playing a video on the following display sizes:

1. 640x480.
2. 1280x720.
3. 1600x1200.
4. 1920x1080.

Which of the following list of test cases is a result of applying the Equivalence Partitioning test technique to test this requirement?

a) Verify that the application can play a video on a display of size 1920x1080 (1 test).
b) Verify that the application can play a video on a display of size 640x480 and 1920x1080 (2 tests).
c) Verify that the application can play a video on each of the display sizes in the requirement (4 tests).
d) Verify that the application can play a video on any one of the display sizes in the requirement (1 test).

Select one option.

FL-4.2.1 (K3) Apply Equivalence Partitioning technique to derive test cases from given requirements

Justification

a) Not correct – See c).
b) Not correct – See c).
c) **Correct** – This is a case where the requirement gives an enumeration of discrete values. Each enumeration value is an Equivalence Class by itself, therefore each will be tested when using Equivalent Partitioning test technique.
d) Not correct – See c).

Question #30 (1 Point)

Which of the following BEST describes how tasks are divided between the test manager and the tester?

a) The test manager plans testing activities and chooses the standards to be followed, while the tester chooses the tools and controls to be used.
b) The test manager plans, organizes, and controls the testing activities, while the tester specifies and executes tests.
c) The test manager plans, monitors, and controls the testing activities, while the tester designs tests and decides about automation frameworks.
d) The test manager plans and organizes the testing and specifies the test cases, while the tester prioritizes and executes the tests.

Select one option.

FL-5.1.2 (K1) Identify the tasks of a test manager and tester

Justification

a) Not correct – Syllabus 5.1.2: The tester uses the tools.
b) **Correct** – See Syllabus 5.1.2.
c) Not correct – Deciding about automation frameworks is not a tester's task.
d) Not correct – Test manager does not specify the test cases.

Question #31 (1 Point)

Which of the following metrics would be MOST useful to monitor during test execution?

a) Percentage of executed test cases.
b) Percentage of work done in test environment preparation.
c) Percentage of planned test cases prepared.
d) Percentage of work done in test case preparation.

Select one option.

FL-5.3.1 (K1) Recall metrics used for testing

Justification

a) **Correct** Syllabus 5.3.1: Test case execution (e.g. number of test cases run/not run, and test cases passed/failed).
b) Not correct – Should be monitored during test preparation.
c) Not correct – Should be monitored during test preparation.
d) Not correct – Should be monitored during test preparation.

Question #32 (1 Point)

Which TWO of the following can affect and be part of test planning?

a) Budget limitations.
b) Test objectives.
c) Test log.
d) Failure rate.
e) Use cases.

Select two options.

FL-5.2.1 (K2) Summarize the purpose and content of a test plan

Justification

a) **Correct** – When you are planning the test and there are budget limitations, prioritizing is needed; what should be tested and what should be omitted.
b) **Correct** – See syllabus 5.2.1.
c) Not correct – it is a part of test monitoring and control.
d) Not correct – it is a part of test monitoring and control.
e) Not correct – it is a part of test design.

Question #33 (1 Point)

Which of the following are typical exit criteria from testing?

a) Reliability measures, degree of tester's independence, and product completeness.
b) Reliability measures, test cost, availability of testable code, time to market, and product completeness.
c) Reliability measures, test cost, schedule and unresolved defects.
d) Time to market, residual defects, tester qualification, degree of tester independence and test cost.

Select one option.

FL-5.2.3 (K2) Give examples of potential entry and exit criteria

Justification

a) Not correct – Degree of tester's independence does not play a role in exit criteria.
b) Not correct – "Availability of testable code" is an entry criteria.
c) **Correct** – See Syllabus 5.2.3.
d) Not correct – Degree of tester's independence as well as tester qualification do not play a role in exit criteria.

Question #34 (1 Point)

Which one of the following is **NOT** included in a test summary report?

a) Defining pass/fail criteria and objectives of testing.
b) Deviations from the test approach.
c) Measurements of actual progress against exit criteria.
d) Evaluation of the quality of the test item.

Select one option.

FL-5.3.2 (K2) Summarize the purposes, content, and audiences for test reports

Justification

a) **Correct** – This information has been defined earlier in the test project.
b) Not correct – This information is included in a test report; see the Syllabus chapter 5.3.2: Information on what occurred during a test period.
c) Not correct – This information is included in a test report; see Syllabus 5.3.2: Information and metrics to support recommendations and decisions about future actions, such as an assessment of defects remaining, the economic benefit of continued testing, outstanding risks, and the level of confidence in the tested software.
d) Not correct – This information is included in a test report; see Syllabus 5.3.2: Information and metrics to support recommendations and decisions about future actions, such as an assessment of defects remaining, the economic benefit of continued testing, outstanding risks, and the level of confidence in the tested software.

Question #35 (1 Point)

There are several test strategies. Which strategy (1-4) is characterized by which description (A-D) below?

1. Analytical.
2. Methodical.
3. Model-based.
4. Consultative.

A. Tests are based on a state diagram of a required aspect of the product
B. Tests are designed and prioritized based on the level of risk.
C. Systematic use of some predefined set of test conditions.
D. Tests are chosen based on the views of business domain experts.

a) 1D, 2B, 3A, 4C.
b) 1A, 2C, 3D, 4B.
c) 1D, 2C, 3B, 4A.
d) 1B, 2C, 3A, 4D.

Select one option.

FL-5.2.2 (K2) Differentiate between various test strategies

Justification

Analytical: Syllabus 5.2.2, This type of test strategy is based on an analysis of some factor (e.g. requirement or risk). (1B).

Methodical: Syllabus 5.2.2, In this type of test strategy relies on making systematic use of some predefined set of tests or test conditions, (2C).

Model-based: Syllabus 5.2.2, In this test strategy, tests are designed based on some model of some required aspect of the product, … (3A).

Consultative (or Directed): Syllabus 5.2.2, This type of test strategy is driven primarily by the advice, guidance, or instructions of stakeholders, business domain experts, or technology experts, who may beoutside the test team or outside the organization itself. (4D).

Thus:

a) Not correct.
b) Not correct.
c) Not correct.
d) **Correct.**

Question #36 (1 Point)

Which one of the following is the characteristic of a metrics-based approach for test estimation?

a) Budget which was used by a previous similar test project.
b) Overall experience collected in interviews with test managers.
c) Overall estimate agreed with the developers.
d) Average of calculations collected from business experts.

Select one option.

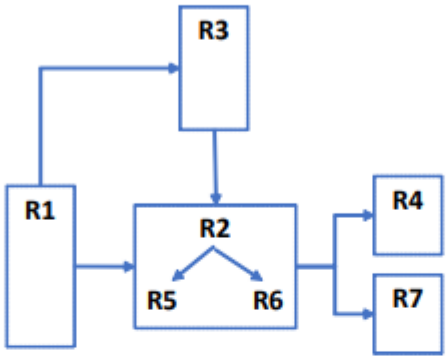
FL-5.2.6 (K2) Explain the difference between two estimation techniques: the metrics-based technique and the expert-based technique

Justification

a) **Correct** – From Syllabus chapter 5.2.6: The metrics-based approach: estimating the testing effort based on metrics of former similar projects or based on typical values .
b) Not correct – This is expert-based approach: estimating the tasks based on estimates made by the owners of the tasks or by experts.
c) Not correct – This is expert-based approach: estimating the tasks based on estimates made by the owners of the tasks or by experts.
d) Not correct – This is expert-based approach: estimating the tasks based on estimates made by the owners of the tasks or by experts.

Question #37 (1 Point)

The following diagram shows the logical dependencies between a set of seven requirements, where a dependency is shown by an arrow. For example, "R1 -> R3" means that R3 depends on R1.



Which one of the following options structures the test execution schedule according to the requirement dependencies?

a) R1 → R3 → R1 → R2 → R5 → R6 → R4 → R7.
b) R1 → R3 → R2 → R5 → R2 → R6 → R4 → R7.
c) R1 → R3 → R2 → R5 → R6 → R4 → R7.
d) R1 → R2 → R5 → R6 → R3 → R4 → R7.

Select one option.

FL-5.2.4 (K3) Apply knowledge of prioritization, technical and logical dependencies to schedule test execution for a given set of test cases

Justification

a) Not correct – everything is dependent on R1, so any test flow that does not start with R1 is FALSE.
b) Not correct – everything is dependent on R1, so any test flow that does not start with R1 is FALSE.
c) **Correct** – the tests are specified in a sequence that takes the dependencies into account.
d) Not correct – R2 is dependent on R3, so R3 should be tested before R2.

Question #38 (1 Point)

You are testing a new version of software for a coffee machine. The machine can prepare different types of coffee based on four categories. i.e. coffee size, sugar, milk and syrup. The criteria are as follows:

- Coffee size (small, medium, large),
- Sugar (none, 1 unit, 2 units, 3 units, 4 units),
- Milk (yes or no),
- Coffee flavor syrup (no syrup, caramel, hazelnut, vanilla).

Now you are writing a defect report with the following information:

Title: Low coffee temperature.
Short summary: When you select coffee with milk, the time for preparing coffee is too long and the temperature of the beverage is too low (less than 40 °C)
Expected result: The temperature of coffee should be standard (about 75 °C).
Degree of risk: Medium
Priority: Normal

What valuable information is MOST likely to be omitted in the above defect report?

a) The actual test result.
b) Data identifying the tested coffee machine.
c) Status of the defect.
d) Ideas for improving the test case.

Select one option.

FL-5.6.1 (K3) Write a defect report, covering defects found during testing.

Justification

a) Not correct – the test result is given in the short summary.
b) **Correct** – when testing different versions of software, identifying information is necessary. Syllabus 5.6: Identification of the test item (configuration item being tested) and environment.
c) Not correct – You are just writing the defect report, hence the status is automatically open.
d) Not correct – This information is useful for the tester, but does not need to be included in the defect report.

Question #39 (1 Point)

Which one of the following is MOST likely to be a benefit of using test execution tools?

a) It is easy to create regression tests.
b) It is easy to maintain version control of test assets.
c) It is easy to design tests for security testing.
d) It is easy to run regression tests.

Select one option.

FL-6.1.2 (K1) Identify benefits and risks of test automation

Justification

a) Not correct – The benefits are not when creating regressions tests, more in executing them.
b) Not correct – This is done by configuration Management tools.
c) Not correct – This needs specialized tools.
d) **Correct** – Syllabus 6.1.2: Reduction in repetitive manual work (e.g. running regression tests, environment set up/tear down tasks, re-entering the same test data, and checking against coding standards), thus saving time.

Question #40 (1 Point)

Which test tool is characterized by the classification below?

1. Tool support for management of testing and testware.
2. Tool support for static testing.
3. Tool support for test execution and logging.
4. Tool support for performance measurement and dynamic analysis.

A. Coverage tools.
B. Configuration management tools.
C. Review tools.
D. Monitoring tools.

a) 1A, 2B, 3D, 4C.
b) 1B, 2C, 3D, 4A.
c) 1A, 2C, 3D, 4B.
d) 1B, 2C, 3A, 4D.

Select one option.

FL-6.1.1 (K2) Classify test tools according to their purpose and the test activities they support

Justification

Support for management of testing and testware: Syllabus 6.1.1, Configuration management tools, (1B).

Support for static testing: Syllabus 6.1.1, Tools that support reviews, (2C).

Support for test execution and logging: Syllabus 6.1.1, Coverage tools, (3A).

Support for performance measurement and dynamic analysis: Syllabus 6.1.1, Performance testing tools/monitoring tools/dynamic analysis tools, (4D).

Thus:

a) Not correct
b) Not correct
c) Not correct
d) **Correct**