CTFL V4.0 Chapter-1 Practice Question Answers Set-2

Question 1

Why is it important to take the “tests wear out” principle into account while performing testing?

a) Running the same tests repeatedly increases the likelihood of finding defects.

Explanation

Running the same tests repeatedly doesn't increase the likelihood of finding defects. In reality, running the same tests repeatedly decreases the likelihood of finding defects.

b) Running the same tests over and over reduces the chance of finding new defects. (**Correct answer**)

Explanation

The "tests wear out" principle emphasizes that running the same tests over and over reduces the chance of finding new defects. This is because as tests are repeatedly executed, they become less effective at uncovering additional defects, leading to diminishing returns in the testing process.

c) Tests should not be context dependent.

Explanation

Tests should be context dependent, so this sentence is not correct. Also, this choice does not address the concept of the "tests wear out" principle.

d) Dynamic tests wear out quicker than static tests.

Explanation

The statement that dynamic tests wear out quicker than static tests is meaningless.
Overall explanation
As per syllabus:
1.3 Testing Principles
A number of testing principles offering general guidelines applicable to all testing have been suggested over the years. This syllabus describes seven such principles.
5. Tests wear out. If the same tests are repeated many times, they become increasingly ineffective in detecting new defects. To overcome this effect, existing tests and test data may need to be modified, and new tests may need to be written. However, in some cases, repeating the same tests can have a beneficial outcome, e.g., in automated regression testing.
Domain
1.3 Testing Principles

Question 2

What definition BEST describes Testware:

a) Work products produced during the test process for use in planning, designing, executing, evaluating and reporting on testing. (**Correct answer)**

Explanation

Testware refers to all the artifacts created and used during the testing process, including test plans, test cases, test scripts, test data, test reports, and any other documentation related to testing activities. These work products are essential for planning, designing, executing, evaluating, and reporting on testing.

b) The body of knowledge used as the basis for test analysis and design.

Explanation

This is the definition of Test Basis

c) A part of a test object used in the test process.

Explanation

This is the definition of the test item

d) None of the above.

Explanation

First choice correctly identified Testware
Overall explanation
As per Glossary:
Testware: Work products produced during the test process for use in planning, designing, executing, evaluating and reporting on testing.
Test Basis: The body of knowledge used as the basis for test analysis and design.
Test item: A part of a test object used in the test process.
Domain
1.4 Test Activities, Testware and Test Roles

Question 3

Which of the following is NOT a benefit of good traceability?

a) It helps to ensure that all requirements are covered by tests.

Explanation

Good traceability ensures that all requirements are covered by tests, allowing for comprehensive test coverage and validation of the system against the specified requirements. This benefit helps in identifying any gaps in testing and ensures that the system meets the intended functionality.

b) It facilitates the identification of the impact of changes.

Explanation

Traceability facilitates the identification of the impact of changes by linking requirements to test cases and defects. This allows for better understanding of how changes in one part of the system may affect other areas, enabling more effective risk assessment and change management.

c) It makes it easier to communicate the results of testing to stakeholders.

Explanation

Traceability makes it easier to communicate the results of testing to stakeholders by providing a clear link between the tests performed and the requirements being tested. This transparency helps stakeholders understand the testing process, results, and any potential risks or issues identified during testing.

d) It adds a layer of difficulty to testing. (**Correct answer**)

Explanation

Good traceability does not add a layer of difficulty to testing; instead, it enhances the testing process by providing a clear link between requirements, test cases, and defects. It helps in ensuring comprehensive test coverage, identifying the impact of changes, and communicating testing results effectively to stakeholders.

Overall explanation

As per Syllabus:

1.4.4. Traceability between the Test Basis and Testware

Accurate traceability supports coverage evaluation, so it is very useful if measurable coverage criteria are defined in the test basis. The coverage criteria can function as key performance indicators to drive the activities that show to what extent the test objectives have been achieved.

For example:

- Traceability of test cases to requirements can verify that the requirements are covered by test cases.
- Traceability of test results to risks can be used to evaluate the level of residual risk in a test object.

In addition to evaluating coverage, good traceability makes it possible to determine the impact of changes, facilitates test audits, and helps meet IT governance criteria. Good traceability also makes test progress and completion reports more easily understandable by including the status of test basis elements. This can also assist in communicating the technical aspects of testing to stakeholders in an understandable manner. Traceability provides information to assess product quality, process capability, and project progress against business goals.

Domain

1.4 Test Activities, Testware and Test Roles

Question 4

Match the following test activities with their corresponding work products.

1. Test planning
2. Test design

3. Test Implementation
4. Test Completion
A. Test execution schedule
B. Change requests (e.g., as product backlog items).
C. Test schedule
D. Test charters

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

Explanation

Test planning involves creating a test schedule to outline the timeline for testing activities. This includes defining when different test activities will take place, such as test execution, review, and reporting. Therefore, the correct match for test planning is 1C.

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

Explanation

Test design is the process of creating test charters, which are documents outlining the scope, objectives, and approach of testing activities. Test charters help testers understand what needs to be tested and how to approach the testing process. Therefore, the correct match for test design is 2D.

c) 1C, 2D, 3A, 4B. (**Correct answer)**

Explanation

Test Implementation involves creating test execution schedules, which outline when specific tests will be executed and by whom. This work product helps ensure that testing activities are carried out efficiently and effectively. Therefore, the correct match for test implementation is 3A.

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

Explanation

Test planning involves creating a test schedule to outline the timeline for testing activities. This includes defining when different test activities will take place, such as test execution, review, and reporting. Therefore, the correct match for test planning is 1C.

Overall explanation

1.4.3 Testware

Testware is created as output work products from the test activities. There is a significant variation in how different organizations produce, shape, name, organize and manage their work products. Proper configuration management ensures consistency and integrity of work products. The following list of work products is not exhaustive:

- Test planning work products include: test plan, test schedule, risk register, and entry and exit criteria. Risk register is a list of risks together with risk likelihood, risk impact and information about risk mitigation. Test schedule, risk register and entry and exit criteria are often a part of the test plan.
- Test monitoring and control work products include: test progress reports, documentation of control directives and risk information.

- Test analysis work products include: (prioritized) test conditions (e.g., acceptance criteria), and defect reports regarding defects in the test basis (if not fixed directly).
- Test design work products include: (prioritized) test cases, test charters, coverage items, test data requirements and test environment requirements.
- Test implementation work products include: test procedures, automated test scripts, test suites, test data, test execution schedule, and test environment elements. Examples of test environment elements include: stubs, drivers, simulators, and service virtualizations.
- Test execution work products include: test logs, and defect reports.
- Test completion work products include: test completion report, action items for improvement of subsequent projects or iterations, documented lessons learned, and change requests (e.g., as product backlog items).

Domain
1.4 Test Activities, Testware and Test Roles

Question 5
Which of the following is NOT a necessary skill for the tester to have?

a) Thoroughness and communication skills

Explanation

Thoroughness and communication skills are essential for a tester to effectively communicate issues, report bugs, and ensure thorough testing coverage. These skills help in maintaining clear communication with the development team and other stakeholders, leading to better collaboration and understanding of the testing process.

b) Technical knowledge

Explanation

Technical knowledge is crucial for a tester to understand the software being tested, identify potential issues, and effectively execute test cases. It helps in troubleshooting technical issues, understanding system architecture, and utilizing testing tools efficiently to ensure comprehensive testing coverage.

c) Time management skills (**Correct answer)**

Explanation

Time management skills, while important in various aspects of work, are not a necessary skill specifically for a tester. While testers need to adhere to project timelines and deadlines, time management skills are not unique to the testing role and can be developed and improved over time.

d) Domain knowledge

Explanation

Domain knowledge is valuable for a tester to understand the industry, business processes, and user expectations related to the software being tested. It helps in identifying critical scenarios, prioritizing test cases, and providing valuable insights during testing.

Overall explanation
As per syllabus page 21:
1.5.1. Generic Skills Required for Testing
While being generic, the following skills are particularly relevant for testers:
• Testing knowledge (to increase effectiveness of testing, e.g., by using test techniques)
• Thoroughness, carefulness, curiosity, attention to details, being methodical (to identify defects, especially the ones that are difficult to find)
• Good communication skills, active listening, being a team player (to interact effectively with all stakeholders, to convey information to others, to be understood, and to report and discuss defects)
• Analytical thinking, critical thinking, creativity (to increase effectiveness of testing)
• Technical knowledge (to increase efficiency of testing, e.g., by using appropriate test tools)
• Domain knowledge (to be able to understand and to communicate with end users/business representatives)
Domain
1.5 Essential Skills and Good Practices in Testing

Question 6
Which of the following is NOT a contextual factor that can impact the way testing is carried out?

a) Availability of team members.

Explanation

The availability of team members can impact the way testing is carried out as it determines the resources and skill sets available for testing activities. A lack of team members can lead to delays or inefficiencies in testing processes.

b) Availability of tools.

Explanation

The availability of tools can significantly impact testing activities as the right tools can streamline and automate testing processes, leading to more efficient and effective testing. Lack of access to necessary tools can hinder testing efforts.

c) Business domain.

Explanation

The business domain in which the software operates can greatly influence testing strategies and priorities. Understanding the specific requirements and challenges of the business domain is crucial for designing relevant and effective test cases.

d) The number of defects detected in previous projects. (**Correct answer**)

Explanation

The number of defects detected in previous projects is not a contextual factor that directly impacts the way testing is carried out. While past project data can inform testing strategies, the

presence or absence of defects in previous projects does not dictate how testing should be conducted in the current project.
Overall explanation
As per syllabus page 18:
Test Process in Context
Testing is not performed in isolation. Test activities are an integral part of the development processes carried out within an organization. Testing is also funded by stakeholders and its final goal is to help fulfill the stakeholders' business needs. Therefore, the way the testing is carried out will depend on a number of contextual factors including:
• Stakeholders (needs, expectations, requirements, willingness to cooperate, etc.)
• Team members (skills, knowledge, level of experience, availability, training needs, etc.)
• Business domain (criticality of the test object, identified risks, market needs, specific legal regulations, etc.)
• Technical factors (type of software, product architecture, technology used, etc.)
• Project constraints (scope, time, budget, resources, etc.)
• Organizational factors (organizational structure, existing policies, practices used, etc.)
• Software development lifecycle (engineering practices, development methods, etc.)
• Tools (availability, usability, compliance, etc.)
Domain
1.4 Test Activities, Testware and Test Roles

Question 7
A software developer is working on a new feature for a web application. The feature allows users to upload images to the application. The developer writes the code to upload the images, but they forget to include code to check the size of the images. As a result, users can upload images that are too large, which causes the application to crash. Which of the following is the error?

a) The Developer forgetting to add code to check the image size. (**Correct answer**)

Explanation

The error (mistake) in this scenario is the developer forgetting to add code to check the size of the images. This omission leads to users being able to upload images that are too large, causing the application to crash. Implementing size checks is essential to prevent such issues and ensure the stability of the application.

b) The application crashing

Explanation

While the application crashing is a consequence of the error, aka a failure the root cause lies in the developer forgetting to include code to check the size of the images. The crash occurs due to the lack of validation for image sizes, highlighting the importance of implementing proper error handling and input validation in software development.

c) The users uploading images with sizes that aren't supported by the website.

Explanation

Users uploading images with sizes that aren't supported by the website is a symptom of the underlying issue, which is the developer's oversight in implementing image size checks. Without proper validation mechanisms in place, users can inadvertently upload images that exceed the application's capacity, leading to crashes and potential data loss.

d) None of the above.

Explanation

None of the above is not the correct choice in this scenario, as the error stems from the developer's failure to incorporate image size checks in the code. Addressing this oversight is crucial to prevent application crashes and ensure a seamless user experience when uploading images.

Overall explanation

As per syllabus page 17:

1.2.3. Errors, Defects, Failures, and Root Causes

Human beings make errors (mistakes), which produce defects (faults, bugs), which in turn may result in failures. Humans make errors for various reasons, such as time pressure, complexity of work products, processes, infrastructure or interactions, or simply because they are tired or lack adequate training. Defects can be found in documentation, such as a requirements specification or a test script, in source code, or in a supporting artifact such as a build file. Defects in artifacts produced earlier in the SDLC, if undetected, often lead to defective artifacts later in the lifecycle. If a defect in code is executed, the system may fail to do what it should do, or do something it shouldn’t, causing a failure. Some defects will always result in a failure if executed, while others will only result in a failure in specific circumstances, and some may never result in a failure. Errors and defects are not the only cause of failures. Failures can also be caused by environmental conditions, such as when radiation or electromagnetic field cause defects in firmware. A root cause is a fundamental reason for the occurrence of a problem (e.g., a situation that leads to an error). Root causes are identified through root cause analysis, which is typically performed when a failure occurs or a defect is identified. It is believed that further similar failures or defects can be prevented or their frequency reduced by addressing the root cause, such as by removing it.

Domain

1.2 Why is Testing Necessary?

Question 8

You're a QA engineer testing a new mobile game. For the past few weeks, you've been diligently running a set of automated tests that cover core gameplay mechanics. These tests have successfully identified several bugs related to character movement, item interaction, and level progression. However, lately, the tests haven't uncovered any new defects despite repeated runs. You suspect there might be additional bugs lurking, but the development team is confident that the game is stable based on the passing tests. You believe it's time to revisit the test strategy.

Which of the following principles explains your concern?

a) Testing shows the presence, not the absence, of defects.

Explanation

While this principle is generally true, it doesn't directly address your specific concern. You've already found defects, but you're worried the current tests might be missing some.

b) Exhaustive testing is impossible.

Explanation

This principle emphasizes the impracticality of testing everything. While relevant for test design, it doesn't explain why the existing tests might be less effective over time.

c) Tests wear out (**Correct answer)**

Explanation

This principle perfectly explains your concern. Repeatedly running the same tests can make them less effective at finding new defects. Your suspicion that additional bugs exist despite passing tests aligns with this principle.

d) Defects cluster together.

Explanation

In the scenario, you've already identified defects related to core gameplay mechanics, which could be indicative of clustering in that area. However, your concern isn't about identifying the location of defects (which might already be happening). You suspect the existing tests, designed to cover those core mechanics, are becoming ineffective at finding new issues within the same area.

Overall explanation

As per syllabus page 17 & 18:

1.3.Testing Principles

A number of testing principles offering general guidelines applicable to all testing have been suggested over the years. This syllabus describes seven such principles.

1. Testing shows the presence, not the absence of defects. Testing can show that defects are present in the test object, but cannot prove that there are no defects. Testing reduces the probability of defects remaining undiscovered in the test object, but even if no defects are found, testing cannot prove test object correctness.

2. Exhaustive testing is impossible. Testing everything is not feasible except in trivial cases. Rather than attempting to test exhaustively, test techniques, test case prioritization, and risk-based testing, should be used to focus test efforts.

3. Early testing saves time and money. Defects that are removed early in the process will not cause subsequent defects in derived work products. The cost of quality will be reduced since fewer failures will occur later in the SDLC. To find defects early, both static testing and dynamic testing should be started as early as possible.

4. Defects cluster together. A small number of system components usually contain most of the defects discovered or are responsible for most of the operational failures. This phenomenon is

an illustration of the Pareto principle. Predicted defect clusters, and actual defect clusters observed during testing or in operation, are an important input for risk-based testing.
5. Tests wear out. If the same tests are repeated many times, they become increasingly ineffective in detecting new defects. To overcome this effect, existing tests and test data may need to be modified, and new tests may need to be written. However, in some cases, repeating the same tests can have a beneficial outcome, e.g., in automated regression testing.
6. Testing is context dependent. There is no single universally applicable approach to testing. Testing is done differently in different contexts.
7. Absence-of-defects fallacy. It is a fallacy (i.e., a misconception) to expect that software verification will ensure the success of a system. Thoroughly testing all the specified requirements and fixing all the defects found could still produce a system that does not fulfill the users' needs and expectations, that does not help in achieving the customer's business goals, and that is inferior compared to other competing systems. In addition to verification, validation should also be carried out.

Question 9

Which of the following options shows an example of test activities that contribute to success?

a) Having testers involved during various software development lifecycle (SDLC) activities will help to detect defects in work products. (**Correct answer**)

Explanation

Having testers involved throughout the software development lifecycle (SDLC) activities allows for early detection of defects in work products. This proactive approach helps in identifying issues sooner, leading to better quality software in the end.

b) Testers minimizing interruptions to developers during coding, allowing them to focus.

Explanation

While a peaceful coding environment might be ideal, it doesn't directly address testing's contribution to success.

c) Testers actively participating in user forums to understand emerging user trends helps identify potential future product features.

Explanation

While actively participating in user forums to understand emerging user trends can be beneficial for product development, it may not directly contribute to the success of test activities. Understanding user trends can help in identifying potential future product features, but it may not necessarily improve the quality of testing or lead to successful test outcomes.

d) Certified testers will design much better test cases than non-certified testers.

Explanation

Being a certified tester does not guarantee the design of much better test cases compared to non-certified testers. Test case design skills are developed through experience, knowledge of

testing techniques, and understanding of the system under test. Certification may indicate a certain level of knowledge, but it does not automatically translate to better test case design.

Question 10
You have been assigned as a tester to a team producing a new system incrementally. You have noticed that no changes have been made to the existing regression test cases for several iterations and no new regression defects were identified. Your manager is happy, but you are not.
Which testing principle explains your skepticism?

a) Tests wear out. (**Correct answer)**

Explanation

The testing principle "Tests wear out" suggests that over time, the effectiveness of test cases diminishes as they become less capable of identifying new defects. In this scenario, the lack of changes to existing regression test cases over several iterations may indicate that these tests have become less effective at uncovering new regression defects, leading to skepticism about the thoroughness of the testing process.

b) Absence-of-errors is a fallacy.

Explanation

The absence-of-errors fallacy is not applicable in this scenario as it refers to the misconception that the absence of identified defects in a system indicates its readiness for release. In this case, the skepticism arises from the lack of changes to regression test cases and the absence of new regression defects, rather than a false belief in the system's defect-free state.

c) Defects cluster together.

Explanation

The principle of "Defects cluster together" suggests that defects tend to occur in specific areas of a system or codebase. While this principle may be relevant in identifying patterns of defects, it does not directly explain the skepticism towards the unchanged regression test cases and lack of new regression defects in this particular situation.
d) Exhaustive testing is impossible.

Explanation

The principle that "Exhaustive testing is impossible" highlights the fact that it is not feasible to test every possible combination of inputs and scenarios in a system. While this principle may contribute to the skepticism about the testing process, it does not specifically address the concern related to the lack of changes in regression test cases and the absence of new regression defects over multiple iterations.
Overall explanation
As per syllabus page 17 & 18:
1.3.Testing Principles

A number of testing principles offering general guidelines applicable to all testing have been suggested over the years. This syllabus describes seven such principles.
1. Testing shows the presence, not the absence of defects. Testing can show that defects are present in the test object, but cannot prove that there are no defects. Testing reduces the probability of defects remaining undiscovered in the test object, but even if no defects are found, testing cannot prove test object correctness.
2. Exhaustive testing is impossible. Testing everything is not feasible except in trivial cases. Rather than attempting to test exhaustively, test techniques, test case prioritization, and risk-based testing, should be used to focus test efforts.
3. Early testing saves time and money. Defects that are removed early in the process will not cause subsequent defects in derived work products. The cost of quality will be reduced since fewer failures will occur later in the SDLC. To find defects early, both static testing and dynamic testing should be started as early as possible.
4. Defects cluster together. A small number of system components usually contain most of the defects discovered or are responsible for most of the operational failures. This phenomenon is an illustration of the Pareto principle. Predicted defect clusters, and actual defect clusters observed during testing or in operation, are an important input for risk-based testing.
5. Tests wear out. If the same tests are repeated many times, they become increasingly ineffective in detecting new defects. To overcome this effect, existing tests and test data may need to be modified, and new tests may need to be written. However, in some cases, repeating the same tests can have a beneficial outcome, e.g., in automated regression testing.
6. Testing is context dependent. There is no single universally applicable approach to testing. Testing is done differently in different contexts.
7. Absence-of-defects fallacy. It is a fallacy (i.e., a misconception) to expect that software verification will ensure the success of a system. Thoroughly testing all the specified requirements and fixing all the defects found could still produce a system that does not fulfill the users' needs and expectations, that does not help in achieving the customer's business goals, and that is inferior compared to other competing systems. In addition to verification, validation should also be carried out.
Domain
1.3 Testing Principles

Question 11
Which of the following skills (i-v) are the MOST important skills of a tester?
i. Having domain knowledge
ii. Creating a product vision
iii. Being a good team player
iv. Planning and organizing the work of the team
v. Critical thinking
a) ii and iv

Explanation

Creating a product vision and planning and organizing the work of the team are not directly related to the skills of a tester. While these skills may be valuable in certain contexts, they are not essential skills that define a tester's role in software testing.

b) i, iii and v (**Correct answer**)

Explanation

Having domain knowledge, being a team player, and critical thinking are all relevant skills of a tester. Domain knowledge helps in understanding the software being tested, being a team player facilitates collaboration within the testing team, and critical thinking is essential for identifying potential issues and risks in the software.

c) i, ii and v

Explanation

Domain knowledge (i) is crucial for understanding the product being tested, creating a product vision (ii) this is not primarily a tester's responsibility. Testers focus on evaluating the existing product, not defining its overall vision. Critical thinking (v) is essential for identifying potential issues and risks in the testing process.

d) iii and iv

Explanation

Planning and organizing the work of the team are not directly related to the skills of a tester. While this skills may be valuable in certain contexts, it is not an essential skills that defines a tester's role in software testing.

Overall explanation

As per syllabus page 21:

1.5.1. Generic Skills Required for Testing

While being generic, the following skills are particularly relevant for testers:

• Testing knowledge (to increase effectiveness of testing, e.g., by using test techniques)

• Thoroughness, carefulness, curiosity, attention to details, being methodical (to identify defects, especially the ones that are difficult to find)

• Good communication skills, active listening, being a team player (to interact effectively with all stakeholders, to convey information to others, to be understood, and to report and discuss defects)

• Analytical thinking, critical thinking, creativity (to increase effectiveness of testing)

• Technical knowledge (to increase efficiency of testing, e.g., by using appropriate test tools)

• Domain knowledge (to be able to understand and to communicate with end users/business representatives)

Domain

1.5 Essential Skills and Good Practices in Testing

Question 12

Which of the following tasks belong mainly to a testing role?

a) Configure test environments
b) Maintain the product backlog
c) Design solutions to new requirements
d) Create the test plan
e) Report found defects
Select TWO options.

a) A and E (**Correct answer)**

Explanation

Configuring test environments is a task that falls mainly under the testing role as it involves setting up the necessary infrastructure and tools for conducting tests. Reporting on found defects is also a key responsibility of a tester, as it involves documenting and communicating issues discovered during testing to the relevant stakeholders.

b) A and B

Explanation

Maintaining the product backlog is a responsibility of a product owner or a project manager, not a testing role. Reporting on found defects, on the other hand, is a key task for testers as it involves documenting and communicating issues discovered during testing.

c) B and E

Explanation

Maintaining the product backlog is a task that is typically associated with a product owner or a project manager, rather than a testing role. Reporting on achieved coverage, however, is a task that belongs to a testing role, as it involves analyzing and communicating the extent to which testing has been conducted.

d) C and D

Explanation

Designing solutions to new requirements and creating the test plan are tasks that are typically associated with roles such as business analysts, developers, or test managers. These tasks are not primarily part of a testing role, making this choice incorrect.
Overall explanation
As per syllabus page 21:
1.4.5. Roles in Testing
In this syllabus, two principal roles in testing are covered: a test management role and a testing role. The activities and tasks assigned to these two roles depend on factors such as the project and product context, the skills of the people in the roles, and the organization.
The test management role takes overall responsibility for the test process, test team and leadership of the test activities. The test management role is mainly focused on the activities of test planning, test monitoring and control and test completion. The way in which the test management role is carried out varies depending on the context. For example, in Agile software development, some of the test management tasks may be handled by the Agile team. Tasks

that span multiple teams or the entire organization may be performed by test managers outside of the development team.
The testing role takes overall responsibility for the engineering (technical) aspect of testing. The testing role is mainly focused on the activities of test analysis, test design, test implementation and test execution. Different people may take on these roles at different times. For example, the test management role can be performed by a team leader, by a test manager, by a development manager, etc. It is also possible for one person to take on the roles of testing and test management at the same time.
Domain
1.4 Test Activities, Testware and Test Roles

Question 13
[RE] Which of the following options shows an example of test activities that contribute to success?

a) Having testers involved during various software development lifecycle (SDLC) activities will help to detect defects in work products. (Correct answer)
b) Testers try not to disturb the developers while coding, so that the developers write better code.
c) Testers collaborating with end users help to improve the quality of defect reports during component integration and system testing.
d) Certified testers will design much better test cases than non-certified testers.

Question 14
You have been assigned as a tester to a team producing a new e-commerce mobile app. During one of the meetings with the stakeholder he asked if testing the entire application thoroughly is feasible. Which testing principles will you base your answer on?

a) Exhaustive testing is impossible. (**Correct answer)**

Explanation

The testing principle of "Exhaustive testing is impossible" states that it is not possible to test every possible combination of inputs and actions in a software application. Therefore, testing the entire application thoroughly is not feasible due to the infinite number of test cases that could be generated.

b) Tests wear out.

Explanation

The testing principle of "Tests wear out" suggests that the effectiveness of a test decreases over time as it is repeated. This principle is not directly related to the feasibility of testing the entire application thoroughly, so it is not the basis for the answer to the stakeholder's question.

c) Absence-of-errors fallacy.

Explanation

The testing principle of "Absence-of-errors fallacy" warns against assuming that the absence of detected defects means the software is defect-free. While this principle highlights the limitations of testing in identifying all defects, it does not directly address the feasibility of testing the entire application thoroughly.

d) Defects cluster together.

Explanation

The testing principle of "Defects cluster together" suggests that defects tend to occur in specific areas or modules of a software application. While this principle can guide test planning and focus on high-risk areas, it does not directly address the feasibility of testing the entire application thoroughly.

Overall explanation

As per syllabus page 17 & 18:

1.3.Testing Principles

A number of testing principles offering general guidelines applicable to all testing have been suggested over the years. This syllabus describes seven such principles.

1. Testing shows the presence, not the absence of defects. Testing can show that defects are present in the test object, but cannot prove that there are no defects. Testing reduces the probability of defects remaining undiscovered in the test object, but even if no defects are found, testing cannot prove test object correctness.

2. Exhaustive testing is impossible. Testing everything is not feasible except in trivial cases. Rather than attempting to test exhaustively, test techniques, test case prioritization, and risk-based testing, should be used to focus test efforts.

3. Early testing saves time and money. Defects that are removed early in the process will not cause subsequent defects in derived work products. The cost of quality will be reduced since fewer failures will occur later in the SDLC. To find defects early, both static testing and dynamic testing should be started as early as possible.

4. Defects cluster together. A small number of system components usually contain most of the defects discovered or are responsible for most of the operational failures. This phenomenon is an illustration of the Pareto principle. Predicted defect clusters, and actual defect clusters observed during testing or in operation, are an important input for risk-based testing.

5. Tests wear out. If the same tests are repeated many times, they become increasingly ineffective in detecting new defects. To overcome this effect, existing tests and test data may need to be modified, and new tests may need to be written. However, in some cases, repeating the same tests can have a beneficial outcome, e.g., in automated regression testing.

6. Testing is context dependent. There is no single universally applicable approach to testing. Testing is done differently in different contexts.

7. Absence-of-defects fallacy. It is a fallacy (i.e., a misconception) to expect that software verification will ensure the success of a system. Thoroughly testing all the specified requirements and fixing all the defects found could still produce a system that does not fulfill the users' needs and expectations, that does not help in achieving the customer's business goals,

and that is inferior compared to other competing systems. In addition to verification, validation should also be carried out.
Domain
1.3 Testing Principles

Question 15
Which of the following is NOT a part of the debugging process?

a) Reproduction of a failure.

Explanation

Reproduction of a failure is a crucial part of the debugging process as it involves identifying the steps to recreate the issue, which is essential for understanding and resolving the problem.

b) Writing new test cases. (**Correct answer**)

Explanation

Writing new test cases is not typically a part of the debugging process. Debugging focuses on identifying and fixing existing issues in the code, rather than creating new test cases.

c) Diagnosis (finding the root cause).

Explanation

Diagnosis, or finding the root cause of the issue, is a fundamental step in the debugging process. Understanding why the problem occurred is essential for implementing an effective solution.

d) Fixing the cause.

Explanation

Fixing the cause of the issue is the ultimate goal of the debugging process. Once the root cause has been identified, developers can make the necessary changes to resolve the problem and ensure the software functions correctly.
Overall explanation
As per syllabus 16:
1.1.2. Testing and Debugging
Testing and debugging are separate activities. Testing can trigger failures that are caused by defects in the software (dynamic testing) or can directly find defects in the test object (static testing).
When dynamic testing triggers a failure, debugging is concerned with finding causes of this failure (defects), analyzing these causes, and eliminating them. The typical debugging process in this case involves:
• Reproduction of a failure
• Diagnosis (finding the root cause)
• Fixing the cause
Domain
1.1 What is Testing?

Question 16
Which of the following factors (i-v) have SIGNIFICANT influence on the test process?
i. The SDLC.
ii. The number of defects detected in previous projects.
iii. Criticality of the test object
iv. Project scope.
v. The number of certified testers in the organization.

a) i, ii

Explanation

- The Software Development Life Cycle (SDLC) plays a significant role in shaping the test process as it determines when and how testing activities are conducted throughout the project lifecycle.
- The number of defects detected in previous projects may provide insights into the quality of the software but does not have a direct significant influence on the test process.

b) i, iii, iv (**Correct answer**)

Explanation

- The criticality of the test object is a crucial factor that significantly influences the test process, as it determines the level of testing effort and focus required to ensure the quality of the software.
- The project scope also plays a significant role in defining the testing activities and resources needed to meet project objectives.
- The Software Development Life Cycle (SDLC) plays a significant role in shaping the test process as it determines when and how testing activities are conducted throughout the project lifecycle.

c) ii, iv, v

Explanation

The number of defects detected in previous projects and the number of certified testers in the organization may have some influence on the test process, but they are not as significant as the factors mentioned in choices i, iii, and iv.

d) iii, v

Explanation

While he criticality of the test object is a key factor that significantly influences the test process, the number of certified testers in the organization may contribute to the overall testing capability, but it is not as significant as the criticality of the test object in shaping the test process.
Overall explanation
As per syllabus page 19:
1.4.2. Test Process in Context

Testing is not performed in isolation. Test activities are an integral part of the development processes carried out within an organization. Testing is also funded by stakeholders and its final goal is to help fulfill the stakeholders' business needs. Therefore, the way the testing is carried out will depend on a number of contextual factors including:
• Stakeholders (needs, expectations, requirements, willingness to cooperate, etc.)
• Team members (skills, knowledge, level of experience, availability, training needs, etc.)
• Business domain (criticality of the test object, identified risks, market needs, specific legal regulations, etc.)
• Technical factors (type of software, product architecture, technology used, etc.)
• Project constraints (scope, time, budget, resources, etc.)
• Organizational factors (organizational structure, existing policies, practices used, etc.)
• Software development lifecycle (engineering practices, development methods, etc.)
• Tools (availability, usability, compliance, etc.)
These factors will have an impact on many test-related issues, including: test strategy, test techniques used, degree of test automation, required level of coverage, level of detail of test documentation, reporting, etc.
Domain
1.4 Test Activities, Testware and Test Roles

Question 17
Which of the following factors DOES NOT affect the Testing Objectives:

a) The software development lifecycle (SDLC) being followed.

Explanation

The software development lifecycle (SDLC) being followed can impact testing objectives as different SDLC models may have varying testing requirements and priorities. For example, in an agile SDLC, testing may need to be more iterative and continuous compared to a waterfall SDLC.

b) The work product being tested.

Explanation

The work product being tested is a crucial factor that affects testing objectives. The complexity, size, criticality, and quality of the work product will determine the testing scope, depth, and strategies to be employed to ensure thorough testing.

c) The time available to market the product.

Explanation

The time available to market the product can significantly influence testing objectives. Tight deadlines may require prioritizing testing activities, focusing on critical functionalities, and balancing thorough testing with timely product delivery.

d) The level of automation used in testing (**Correct answer**)

Explanation

The level of automation used in testing is not directly related to testing objectives. While automation can significantly impact testing efficiency, coverage, and reliability, it is a testing technique rather than a factor that affects the overall testing objectives.
Overall explanation
As per syllabus page 15:
1.1.1. Test Objectives
The typical test objectives are:
• Evaluating work products such as requirements, user stories, designs, and code
• Triggering failures and finding defects
• Ensuring required coverage of a test object
• Reducing the level of risk of inadequate software quality
• Verifying whether specified requirements have been fulfilled
• Verifying that a test object complies with contractual, legal, and regulatory requirements
• Providing information to stakeholders to allow them to make informed decisions
• Building confidence in the quality of the test object
• Validating whether the test object is complete and works as expected by the stakeholders
Objectives of testing can vary, depending upon the context, which includes the work product being tested, the test level, risks, the software development lifecycle (SDLC) being followed, and factors related to the business context, e.g., corporate structure, competitive considerations, or time to market.
Domain
1.1 What is Testing?

Question 18
You were given a task to analyze and fix causes of failures in a new system to be released. Which activity are you performing?

a) Debugging **(Correct answer)**
b) Explanation

Debugging involves identifying and fixing the causes of failures in a system. It is the process of locating and resolving errors in the code that are causing the system to behave unexpectedly or fail to meet its requirements.

c) Software testing

Explanation

Software testing involves evaluating the functionality of a system to ensure that it meets specified requirements. While testing may uncover failures, the activity described in the question specifically focuses on analyzing and fixing those failures, which is the essence of debugging.

d) Requirement elicitation

Explanation

Requirement elicitation is the process of gathering and documenting requirements from stakeholders to understand what the system should do. It is not directly related to analyzing and fixing causes of failures in a system, which is the focus of debugging.
Defect management

Explanation

Defect management involves tracking, prioritizing, and resolving defects found during testing. While defect management is related to fixing issues in the system, the specific activity described in the question, analyzing and fixing causes of failures in a new system, aligns more closely with the process of debugging.
Overall explanation
As per syllabus 16:
1.1.2. Testing and Debugging
Testing and debugging are separate activities. Testing can trigger failures that are caused by defects in the software (dynamic testing) or can directly find defects in the test object (static testing).
When dynamic testing triggers a failure, debugging is concerned with finding causes of this failure (defects), analyzing these causes, and eliminating them. The typical debugging process in this case involves:
• Reproduction of a failure
• Diagnosis (finding the root cause)
• Fixing the cause
Domain
1.1 What is Testing?

Question 19
Within software development, the terms "Quality Assurance" (QA) and "Testing" are sometimes used interchangeably. How accurate is this practice?

a) Entirely accurate; both terms encompass all activities related to software quality

Explanation

This choice is not entirely accurate. While both Quality Assurance (QA) and Testing contribute to software quality, they do not encompass all activities related to software quality. QA focuses on processes and procedures to ensure quality, while Testing specifically involves checking if something works as intended and finding any issues.

b) Inaccurate; Testing is a more thorough process that focuses on ensuring that software works as intended

Explanation

Describing "Testing" as a more thorough process compared to Quality Assurance (QA) is inaccurate. Both QA and Testing play crucial roles in ensuring software quality, but they focus on different aspects of the development process.

c) Inaccurate; QA is a part of the Testing process **(Correct answer)**

Explanation

It is inaccurate to consider QA as a part of the Testing process. Quality Assurance (QA) focuses on ensuring quality throughout the entire software development lifecycle, while Testing is a specific activity that checks the functionality and performance of the software.

d) Inaccurate; QA ensures quality throughout the entire development process, while testing checks if something works as intended

Explanation

QA involves activities that ensure quality throughout the entire development process, while Testing specifically focuses on verifying if the software functions as intended. They are complementary but distinct components of software quality assurance.
Overall explanation
1.2.2. Testing and Quality Assurance
(QA) While people often use the terms “testing” and “quality assurance” (QA) interchangeably, testing and QA are not the same. Testing is a form of quality control (QC).
QC is a product-oriented, corrective approach that focuses on those activities supporting the achievement of appropriate levels of quality. Testing is a major form of quality control, while others include formal methods (model checking and proof of correctness), simulation and prototyping.
QA is a process-oriented, preventive approach that focuses on the implementation and improvement of processes. It works on the basis that if a good process is followed correctly, then it will generate a good product. QA applies to both the development and testing processes, and is the responsibility of everyone on a project.
Test results are used by QA and QC. In QC they are used to fix defects, while in QA they provide feedback on how well the development and test processes are performing.
Domain
1.2 Why is Testing Necessary?

Question 20
Which of the following is TRUE regarding Quality Control and Quality Assurance:

a) Quality assurance is a product-oriented process, preventive approach. While quality control is a process oriented corrective approach.

Explanation

Quality assurance is actually a process-oriented preventive approach, focusing on improving processes to prevent defects. On the other hand, quality control is a product-oriented corrective approach, focusing on identifying defects in the final product and correcting them.

b) Quality assurance is a process oriented corrective approach, while quality control is a process oriented preventive approach.

Explanation

Quality assurance is actually a process-oriented preventive approach, emphasizing the importance of following good processes to achieve a good product outcome. On the other hand, quality control is a product-oriented corrective approach, focusing on rectifying defects in the product.

c) Quality assurance is a process oriented preventive approach while quality control is a product oriented corrective approach. **(Correct answer)**

Explanation

This statement correctly distinguishes between quality assurance and quality control. Quality assurance is a process-oriented preventive approach that aims to improve processes to prevent defects, while quality control is a product-oriented corrective approach that focuses on identifying and fixing defects in the final product.

d) Quality assurance and quality control are the same and can be used interchangeably.

Explanation

Quality assurance and quality control are not the same and cannot be used interchangeably. They are two distinct processes with different focuses and objectives in ensuring product quality.

Overall explanation

1.2.2. Testing and Quality Assurance

(QA) While people often use the terms “testing” and “quality assurance” (QA) interchangeably, testing and QA are not the same. Testing is a form of quality control (QC).

QC is a product-oriented, corrective approach that focuses on those activities supporting the achievement of appropriate levels of quality. Testing is a major form of quality control, while others include formal methods (model checking and proof of correctness), simulation and prototyping.

QA is a process-oriented, preventive approach that focuses on the implementation and improvement of processes. It works on the basis that if a good process is followed correctly, then it will generate a good product. QA applies to both the development and testing processes, and is the responsibility of everyone on a project.

Test results are used by QA and QC. In QC they are used to fix defects, while in QA they provide feedback on how well the development and test processes are performing.

Domain

1.2 Why is Testing Necessary?

Question 21

A software developer is reviewing a complex set of user stories for a new feature. Due to the large amount of information, they misunderstand a specific requirement related to data formatting. This misunderstanding leads them to write code that processes the data incorrectly. What best describes an incorrectly implemented data processing code?

a) The root cause

Explanation

The term "root cause" refers to the underlying reason or source of a problem. In this scenario, the root cause would be the misunderstanding of the specific requirement related to data formatting, not the code itself that processes the data incorrectly.

b) A failure

Explanation

A failure typically refers to a deviation from expected behavior or a lack of success in achieving a desired outcome. While the incorrect data processing code may result in a failure, the term "failure" does not accurately describe the code itself.

c) A defect **(Correct answer)**

Explanation

A defect is a flaw or imperfection in a product or system that can lead to incorrect behavior or failure. In this case, the code that processes the data incorrectly due to a misunderstanding of the requirement would be considered a defect.

d) An error

Explanation

An error is a mistake or fault in a program that may cause it to behave unexpectedly or produce incorrect results. While the incorrect data processing code may contain errors, the term "error" does not fully capture the concept of a flaw in the code due to a misunderstanding of the requirement.

Overall explanation

1.2.3. Errors, Defects, Failures, and Root Causes

Human beings make errors (mistakes), which produce defects (faults, bugs), which in turn may result in failures. Humans make errors for various reasons, such as time pressure, complexity of work products, processes, infrastructure or interactions, or simply because they are tired or lack adequate training.

Defects can be found in documentation, such as a requirements specification or a test script, in source code, or in a supporting artifact such as a build file. Defects in artifacts produced earlier in the SDLC, if undetected, often lead to defective artifacts later in the lifecycle. If a defect in code is executed, the system may fail to do what it should do, or do something it shouldn’t, causing a failure. Some defects will always result in a failure if executed, while others will only result in a failure in specific circumstances, and some may never result in a failure.

Errors and defects are not the only cause of failures. Failures can also be caused by environmental conditions, such as when radiation or electromagnetic field cause defects in firmware.

A root cause is a fundamental reason for the occurrence of a problem (e.g., a situation that leads to an error). Root causes are identified through root cause analysis, which is typically performed when a failure occurs or a defect is identified. It is believed that further similar failures or defects can be prevented or their frequency reduced by addressing the root cause, such as by removing it.

Domain
1.2 Why is Testing Necessary?

Question 22
Which of the following is the BEST example of how traceability supports testing?

a) Performing the impact analysis of a change will give information about the completion of the tests.

Explanation

Performing impact analysis of a change is important for understanding the potential effects of the change on the system, but it does not directly relate to how traceability supports testing. It focuses on identifying the areas that may be impacted by the change, rather than the relationship between test cases and test results.

b) Analyzing the traceability between test cases and test results will give information about the estimated level of residual risk.

Explanation

Analyzing the traceability between test cases and test results is crucial for understanding the effectiveness of the testing process and identifying any gaps or discrepancies. However, it does not specifically address how traceability supports testing in terms of selecting the right test cases or improving the testing process.

c) Performing the impact analysis of a change will help selecting the right test cases for regression testing. **(Correct answer)**

Explanation

Performing impact analysis of a change helps in understanding the potential impact on the system and aids in selecting the right test cases for regression testing. This is a key aspect of how traceability supports testing, as it ensures that the regression test suite covers the areas most likely affected by the change.

d) Analyzing the traceability between the test basis, the test objects and the test cases will help in selecting test data to achieve the assumed coverage of the test object.

Explanation

Analyzing the traceability between the test basis, test objects, and test cases is essential for ensuring that the selected test data covers the assumed coverage of the test object. While this is important for test data selection, it does not directly address how traceability supports testing in terms of impact analysis or regression testing selection.
Overall explanation
1.4.4. Traceability between the Test Basis and Testware
In order to implement effective test monitoring and control, it is important to establish and maintain traceability throughout the test process between the test basis elements, testware associated with these elements (e.g., test conditions, risks, test cases), test results, and detected defects.

Accurate traceability supports coverage evaluation, so it is very useful if measurable coverage criteria are defined in the test basis. The coverage criteria can function as key performance indicators to drive the activities that show to what extent the test objectives have been achieved. For example:

• Traceability of test cases to requirements can verify that the requirements are covered by test cases.

• Traceability of test results to risks can be used to evaluate the level of residual risk in a test object.

In addition to evaluating coverage, good traceability makes it possible to determine the impact of changes, facilitates test audits, and helps meet IT governance criteria. Good traceability also makes test progress and completion reports more easily understandable by including the status of test basis elements. This can also assist in communicating the technical aspects of testing to stakeholders in an understandable manner. Traceability provides information to assess product quality, process capability, and project progress against business goals.

Domain

1.4 Test Activities, Testware and Test Roles

Question 23

Which of the following is TRUE regarding the whole team approach:

a) The whole team approach is most useful when used in safety critical systems.

Explanation

The whole team approach is not specifically limited to safety critical systems. It is a general approach that involves collaboration and shared responsibility for quality across all team members, regardless of the system's criticality.

b) In the whole team approach, while everyone is responsible for quality. Every member of the team only does tasks related to his role.

Explanation

In the whole team approach, every member of the team is indeed responsible for quality, but this does not mean that they only do tasks related to their role. The approach encourages collaboration and cross-functional involvement in testing activities.

c) The whole team approach is generally better than independent testing

Explanation

The statement that the whole team approach is generally better than independent testing is not entirely accurate. Both approaches have their own advantages and drawbacks, and the effectiveness of each approach can vary depending on the project and team dynamics.

d) In the whole team approach, developers and testers work closely to agree on the test strategy and decide on test automation approaches. **(Correct answer)**

Explanation

In the whole team approach, developers and testers indeed work closely together to define the test strategy and determine the best approaches for test automation. This collaboration helps ensure that testing activities are aligned with development goals and priorities.
Overall explanation
1.5.2. Whole Team Approach
One of the important skills for a tester is the ability to work effectively in a team context and to contribute positively to the team goals. The whole team approach – a practice coming from Extreme Programming– builds upon this skill.
In the whole-team approach any team member with the necessary knowledge and skills can perform any task, and everyone is responsible for quality. The team members share the same workspace (physical or virtual), as co-location facilitates communication and interaction. The whole team approach improves team dynamics, enhances communication and collaboration within the team, and creates synergy by allowing the various skill sets within the team to be leveraged for the benefit of the project.
Testers work closely with other team members to ensure that the desired quality levels are achieved. This includes collaborating with business representatives to help them create suitable acceptance tests and working with developers to agree on the test strategy and decide on test automation approaches. Testers can thus transfer testing knowledge to other team members and influence the development of the product.
Depending on the context, the whole team approach may not always be appropriate. For instance, in some situations, such as safety-critical, a high level of test independence may be needed.
Domain
1.5 Essential Skills and Good Practices in Testing

Question 24
In software development, what is the primary advantage of having testers with a high degree of independence from the development team?

a) Improved team morale

Explanation

Improved team morale is not the primary advantage of having testers with a high degree of independence from the development team. While it can contribute to a positive work environment, the main focus of independent testers is on ensuring the quality and reliability of the software through unbiased testing processes.

b) Reduced need for training and knowledge transfer

Explanation

Testers with independence from the development team may still require training and knowledge transfer to understand the project requirements, technology stack, and testing processes. Independence does not eliminate the need for training and knowledge sharing, as testers need to be well-equipped to perform their testing tasks effectively.

c) Reduced risk of overlooking defects due to confirmation bias **(Correct answer)**

Explanation

Testers with a high degree of independence from the development team are less likely to overlook defects due to confirmation bias. Confirmation bias occurs when individuals unconsciously seek out information that confirms their existing beliefs or assumptions. Testers who are independent can provide unbiased perspectives and thorough testing, reducing the risk of overlooking defects.

d) Streamlined communication channels

Explanation

While streamlined communication channels are important for effective collaboration between testers and developers, having testers with independence from the development team may not directly impact communication channels. Independence can lead to clearer reporting of defects and findings, but it is not the primary factor in streamlining communication channels within a software development team.

Overall explanation

1.5.3. Independence of Testing

A certain degree of independence makes the tester more effective at finding defects due to differences between the author's and the tester's cognitive biases. Independence is not, however, a replacement for familiarity, e.g., developers can efficiently find many defects in their own code.

Work products can be tested by their author (no independence), by the author's peers from the same team (some independence), by testers from outside the author's team but within the organization (high independence), or by testers from outside the organization (very high independence). For most projects, it is usually best to carry out testing with multiple levels of independence (e.g., developers performing component and component integration testing, test team performing system and system integration testing, and business representatives performing acceptance testing).

The main benefit of independence of testing is that independent testers are likely to recognize different kinds of failures and defects compared to developers because of their different backgrounds, technical perspectives, and biases. Moreover, an independent tester can verify, challenge, or disprove assumptions made by stakeholders during specification and implementation of the system.

However, there are also some drawbacks. Independent testers may be isolated from the development team, which may lead to a lack of collaboration, communication problems, or an adversarial relationship with the development team. Developers may lose a sense of responsibility for quality. Independent testers may be seen as a bottleneck or be blamed for delays in release.

Domain

1.5 Essential Skills and Good Practices in Testing

Question 25

Which of the following belong to the benefits of independent testing?

a. Independent testers may be seen as a bottleneck or be blamed for delays in release.
b. Increased testing coverage.
c. Recognizing different kinds of failures and defects compared to developers.
d. Developers' sense of quality increases because of competition.
e. Independent tester can verify, challenge, or disprove assumptions made by stakeholders.

Select TWO options.

a) A and C.

Explanation

- Independent testers may be seen as a bottleneck or be blamed for delays in release is not a benefit of independent testing, as it can create negative perceptions and hinder the testing process.
- Recognizing different kinds of failures and defects compared to developers is a benefit, as independent testers bring a fresh perspective and can uncover issues that developers might overlook.

b) A and D.

Explanation

- Independent testers may be seen as a bottleneck or be blamed for delays in release is not a benefit of independent testing, as it can create negative perceptions and hinder the testing process.
- Developers' sense of quality increases because of competition is not directly related to the benefits of independent testing, as it focuses on developer motivation rather than the testing process itself.

c) C and E. **(Correct answer)**

Explanation

- Recognizing different kinds of failures and defects compared to developers is a key benefit of independent testing, as it helps in identifying diverse issues and improving overall software quality.
- Independent tester can verify, challenge, or disprove assumptions made by stakeholders is also a benefit, as it ensures that stakeholder expectations are met and potential risks are mitigated.

d) D and E.

Explanation

- Developers' sense of quality increases because of competition is not a benefit of independent testing, as it pertains to developer motivation rather than the testing process itself.
- Independent tester can verify, challenge, or disprove assumptions made by stakeholders is a benefit, as it ensures that stakeholder expectations are met and potential risks are mitigated.

Overall explanation
1.5.3. Independence of Testing
A certain degree of independence makes the tester more effective at finding defects due to differences between the author's and the tester's cognitive biases. Independence is not, however, a replacement for familiarity, e.g., developers can efficiently find many defects in their own code.
Work products can be tested by their author (no independence), by the author's peers from the same team (some independence), by testers from outside the author's team but within the organization (high independence), or by testers from outside the organization (very high independence). For most projects, it is usually best to carry out testing with multiple levels of independence (e.g., developers performing component and component integration testing, test team performing system and system integration testing, and business representatives performing acceptance testing).
The main benefit of independence of testing is that independent testers are likely to recognize different kinds of failures and defects compared to developers because of their different backgrounds, technical perspectives, and biases. Moreover, an independent tester can verify, challenge, or disprove assumptions made by stakeholders during specification and implementation of the system.
However, there are also some drawbacks. Independent testers may be isolated from the development team, which may lead to a lack of collaboration, communication problems, or an adversarial relationship with the development team. Developers may lose a sense of responsibility for quality. Independent testers may be seen as a bottleneck or be blamed for delays in release.
Domain
1.5 Essential Skills and Good Practices in Testing

Question 26
Which of the following is an important objective of testing activities in the software development lifecycle ?

a) Exhaustive testing

Explanation

Exhaustive testing is not a feasible or practical objective in the software development lifecycle. It is impossible to test every possible combination of inputs and scenarios due to time and resource constraints.

b) Ensuring required coverage of a test object **(Correct answer)**

Explanation

Ensuring required coverage of a test object is an important objective of testing activities in the software development lifecycle. It involves identifying and testing all critical aspects of the software to ensure that it meets the specified requirements and functions as intended.

c) Clustering defects

Explanation

Clustering defects is not a primary objective of testing activities in the software development lifecycle. While identifying and categorizing defects is important, the main goal of testing is to ensure the quality and reliability of the software.

d) Debugging

Explanation

Debugging is a separate activity from testing in the software development lifecycle. Testing aims to prevent defects by identifying issues early in the development process, while debugging involves fixing defects that have already been identified.

Overall explanation

As per syllabus page 15:

1.1.1. Test Objectives

The typical test objectives are:

• Evaluating work products such as requirements, user stories, designs, and code
• Triggering failures and finding defects
• Ensuring required coverage of a test object
• Reducing the level of risk of inadequate software quality
• Verifying whether specified requirements have been fulfilled
• Verifying that a test object complies with contractual, legal, and regulatory requirements
• Providing information to stakeholders to allow them to make informed decisions
• Building confidence in the quality of the test object
• Validating whether the test object is complete and works as expected by the stakeholders

Objectives of testing can vary, depending upon the context, which includes the work product being tested, the test level, risks, the software development lifecycle (SDLC) being followed, and factors related to the business context, e.g., corporate structure, competitive considerations, or time to market.

Domain

1.1 What is Testing?

Question 27

When test cases are designed early in the lifecycle, verifying the test basis via the test design, which common test objective is being achieved ?

a) Gaining confidence
b) Finding defects
c) Preventing defects **(Correct answer)**
d) Providing information for decision-making

Overall explanation

Designing test cases early and comparing them (test design) with the test basis (requirements) will help in detecting requirements defects only which will prevent them from escaping to development. In this case, we prevent defects from occurring in the software because they are detected early in the requirements stage before implementing them.

Domain

1.1 What is Testing?

Question 28

Which of the following is an example of debugging ?

a) A tester finds a defect and reports it
b) A tester retests a fix from the developer and finds a regression
c) A developer finds and fixes a defect **(Correct answer)**
d) A developer performs unit testing

Overall explanation

A tester finds a defect and reports it: This is Dynamic Testing

A tester retests a fix from the developer and finds a regression: This is Confirmation testing

A developer finds and fixes a defect: Fixing the defect is debugging no matter who found the defect

A developer performs unit testing: This is unit testing ^_^

==========

According to Syllabus:

Testing & Debugging:

Testing and debugging are different. Executing tests can show failures that are caused by defects in the software. Debugging is the development activity that finds, analyzes, and fixes such defects. Subsequent confirmation testing checks whether the fixes resolved the defects. In some cases, testers are responsible for the initial test and the final confirmation test, while developers do the debugging and associated component testing. However, in Agile development and in some other lifecycles, testers may be involved in debugging and component testing.

==========

According to Glossary:

Debugging: The process of finding, analyzing and removing the causes of failures in software.

Domain

1.1 What is Testing?

Question 29

For which test level may this objective be relevant ?

"Increase code coverage of modules"

a) Component Testing **(Correct answer)**
b) Integration Testing
c) System Testing
d) Acceptance Testing

Overall explanation

According to Syllabus:

1.1.1 Typical Objectives of Testing

During component testing, one objective may be to find as many failures as possible so that the underlying defects are identified and fixed early. Another objective may be to increase code coverage of the component tests.

==========

According to Glossary:
Component Testing [Synonyms: module testing, unit testing]: The testing of individual hardware or software components.
Component Integration Testing [Synonyms: Link Testing]: Testing performed to expose defects in the interfaces and interactions between integrated components.
Integration Testing: Testing performed to expose defects in the interfaces and in the interactions between integrated components or systems.
System Testing: Testing an integrated system to verify that it meets specified requirements.
System Integration Testing: Testing the combination and interaction of systems.
Acceptance Testing: Formal testing with respect to user needs, requirements, and business processes conducted to determine whether or not a system satisfies the acceptance criteria and to enable the user, customers or other authorized entity to determine whether or not to accept the system.
Domain
1.2 Why is Testing Necessary?

Question 30
For which test level may this objective be relevant ?
"Give information to stakeholders about the risk of releasing the system at a given time"

a) Component Testing
b) Integration Testing
c) Functional Testing
d) Acceptance Testing **(Correct answer)**

Overall explanation
According to Syllabus:
1.1.1 Typical Objectives of Testing
During acceptance testing, one objective may be to confirm that the system works as expected and satisfies requirements. Another objective of this testing may be to give information to stakeholders about the risk of releasing the system at a given time.
==========
According to Glossary:
Component Testing [Synonyms: module testing, unit testing]: The testing of individual hardware or software components.
Component Integration Testing [Synonyms: Link Testing]: Testing performed to expose defects in the interfaces and interactions between integrated components.
Integration Testing: Testing performed to expose defects in the interfaces and in the interactions between integrated components or systems.
System Testing: Testing an integrated system to verify that it meets specified requirements.
System Integration Testing: Testing the combination and interaction of systems.
Acceptance Testing: Formal testing with respect to user needs, requirements, and business processes conducted to determine whether or not a system satisfies the acceptance criteria and to enable the user, customers or other authorized entity to determine whether or not to accept the system.

Functional Testing: Testing conducted to evaluate the compliance of a component or system with functional requirements.
Domain
1.2 Why is Testing Necessary?

Question 31
In which Development model may the testers be involved in debugging and component testing ?

a) Agile and iterative models **(Correct answer)**
b) Waterfall Model
c) V-model
d) Mobile Model

Overall explanation
According to Syllabus:
Testing & Debugging:
Testing and debugging are different. Executing tests can show failures that are caused by defects in the software. Debugging is the development activity that finds, analyzes, and fixes such defects. Subsequent confirmation testing checks whether the fixes resolved the defects. In some cases, testers are responsible for the initial test and the final confirmation test, while developers do the debugging and associated component testing. However, in Agile development and in some other lifecycles, testers may be involved in debugging and component testing.
==========
According to Glossary:
Iterative Development Model:
A development lifecycle where a project is broken into a usually large number of iterations. An iteration is a complete development loop resulting in a release (internal or external) of an executable product, a subset of the final product under development, which grows from iteration to iteration to become the final product.
Sequential Development Model:
A type of development lifecycle model in which a complete system is developed in a linear way of several discrete and successive phases with no overlap between them.
V-model:
A sequential development lifecycle model describing a one-for-one relationship between major phases of software development from business requirements specification to delivery, and corresponding test levels from acceptance testing to component testing.
Domain
1.1 What is Testing?

Question 32
Which objective may be obtained by having testers involved in requirements reviews or user story refinement ?

a) Reduces the risk of incorrect or untestable functionality being developed **(Correct answer)**

b) Reduce the risk of fundamental design defects and enable tests to be identified at an early stage
c) Reduce the risk of defects within the code and the tests
d) Increase the likelihood that the software meets stakeholder needs and satisfies requirements

Overall explanation
According to Syllabus:
Testing's Contributions to Success:
Throughout the history of computing, it is quite common for software and systems to be delivered into operation and, due to the presence of defects, to subsequently cause failures or otherwise not meet the stakeholders' needs. However, using appropriate test techniques can reduce the frequency of such problematic deliveries, when those techniques are applied with the appropriate level of test expertise, in the appropriate test levels, and at the appropriate points in the software development lifecycle. Examples include:
-Having testers involved in requirements reviews or user story refinement could detect defects in these work products. The identification and removal of requirements defects reduces the risk of incorrect or untestable functionality being developed.
-Having testers work closely with system designers while the system is being designed can increase each party's understanding of the design and how to test it. This increased understanding can reduce the risk of fundamental design defects and enable tests to be identified at an early stage.
-Having testers work closely with developers while the code is under development can increase each party's understanding of the code and how to test it. This increased understanding can reduce the risk of defects within the code and the tests.
-Having testers verify and validate the software prior to release can detect failures that might otherwise have been missed, and support the process of removing the defects that caused the failures (i.e., debugging). This increases the likelihood that the software meets stakeholder needs and satisfies requirements.
Domain
1.2 Why is Testing Necessary?

Question 33
Which objective may be obtained by having testers work closely with system designers while the system is being designed ?

a) Reduce the risk of incorrect or untestable functionality being developed
b) Reduce the risk of fundamental design defects and enable tests to be identified at an early stage **(Correct answer)**
c) Reduce the risk of defects within the code and the tests
d) Increase the likelihood that the software meets stakeholder needs and satisfies requirements

Overall explanation
According to Syllabus:
Testing's Contributions to Success:

Throughout the history of computing, it is quite common for software and systems to be delivered into operation and, due to the presence of defects, to subsequently cause failures or otherwise not meet the stakeholders' needs. However, using appropriate test techniques can reduce the frequency of such problematic deliveries, when those techniques are applied with the appropriate level of test expertise, in the appropriate test levels, and at the appropriate points in the software development lifecycle. Examples include:
-Having testers involved in requirements reviews or user story refinement could detect defects in these work products. The identification and removal of requirements defects reduces the risk of incorrect or untestable functionality being developed.
-Having testers work closely with system designers while the system is being designed can increase each party's understanding of the design and how to test it. This increased understanding can reduce the risk of fundamental design defects and enable tests to be identified at an early stage.
-Having testers work closely with developers while the code is under development can increase each party's understanding of the code and how to test it. This increased understanding can reduce the risk of defects within the code and the tests.
-Having testers verify and validate the software prior to release can detect failures that might otherwise have been missed, and support the process of removing the defects that caused the failures (i.e., debugging). This increases the likelihood that the software meets stakeholder needs and satisfies requirements.
Domain
1.2 Why is Testing Necessary?
Question 34
Which objective may be obtained by having testers work closely with developers while the code is under development ?

a) Reduce the risk of incorrect or untestable functionality being developed
b) Reduce the risk of fundamental design defects and enable tests to be identified at an early stage
c) Reduce the risk of defects within the code and the tests **(Correct answer)**
d) Increase the likelihood that the software meets stakeholder needs and satisfies requirements

Overall explanation
According to Syllabus:
Testing's Contributions to Success:
Throughout the history of computing, it is quite common for software and systems to be delivered into operation and, due to the presence of defects, to subsequently cause failures or otherwise not meet the stakeholders' needs. However, using appropriate test techniques can reduce the frequency of such problematic deliveries, when those techniques are applied with the appropriate level of test expertise, in the appropriate test levels, and at the appropriate points in the software development lifecycle. Examples include:
-Having testers involved in requirements reviews or user story refinement could detect defects in these work products. The identification and removal of requirements defects reduces the risk of incorrect or untestable functionality being developed.

-Having testers work closely with system designers while the system is being designed can increase each party's understanding of the design and how to test it. This increased understanding can reduce the risk of fundamental design defects and enable tests to be identified at an early stage.
-Having testers work closely with developers while the code is under development can increase each party's understanding of the code and how to test it. This increased understanding can reduce the risk of defects within the code and the tests.
-Having testers verify and validate the software prior to release can detect failures that might otherwise have been missed, and support the process of removing the defects that caused the failures (i.e., debugging). This increases the likelihood that the software meets stakeholder needs and satisfies requirements.
Domain
1.2 Why is Testing Necessary?

Question 35
Which objective may be obtained by having testers verify and validate the software prior to release ?

a) Reduce the risk of incorrect or untestable functionality being developed
b) Reduce the risk of fundamental design defects and enable tests to be identified at an early stage
c) Reduce the risk of defects within the code and the tests
d) Increase the likelihood that the software meets stakeholder needs and satisfies requirements **(Correct answer)**

Overall explanation
According to Syllabus:
Testing's Contributions to Success:
Throughout the history of computing, it is quite common for software and systems to be delivered into operation and, due to the presence of defects, to subsequently cause failures or otherwise not meet the stakeholders' needs. However, using appropriate test techniques can reduce the frequency of such problematic deliveries, when those techniques are applied with the appropriate level of test expertise, in the appropriate test levels, and at the appropriate points in the software development lifecycle. Examples include:
-Having testers involved in requirements reviews or user story refinement could detect defects in these work products. The identification and removal of requirements defects reduces the risk of incorrect or untestable functionality being developed.
-Having testers work closely with system designers while the system is being designed can increase each party's understanding of the design and how to test it. This increased understanding can reduce the risk of fundamental design defects and enable tests to be identified at an early stage.
-Having testers work closely with developers while the code is under development can increase each party's understanding of the code and how to test it. This increased understanding can reduce the risk of defects within the code and the tests.
-Having testers verify and validate the software prior to release can detect failures that might otherwise have been missed, and support the process of removing the defects that caused the

failures (i.e., debugging). This increases the likelihood that the software meets stakeholder needs and satisfies requirements.
Domain
1.2 Why is Testing Necessary?

Question 36
Which of the following is a correct relationship between quality control, quality assurance, and quality management ?

a) Quality management includes both quality control and quality assurance **(Correct answer)**
b) Quality Control includes both quality management and quality assurance
c) Quality Assurance includes both quality management and quality control
d) Quality control and quality assurance are the same and quality management is part of them

Overall explanation
According to Syllabus:
Quality Assurance & Testing:
While people often use the phrase quality assurance (or just QA) to refer to testing, quality assurance and testing are not the same, but they are related. A larger concept, quality management, ties them together.
Quality management includes all activities that direct and control an organization with regard to quality. Among other activities, quality management includes both quality assurance and quality control. Quality assurance is typically focused on adherence to proper processes, in order to provide confidence that the appropriate levels of quality will be achieved. When processes are carried out properly, the work products created by those processes are generally of higher quality, which contributes to defect prevention. In addition, the use of root cause analysis to detect and remove the causes of defects, along with the proper application of the findings of retrospective meetings to improve processes, are important for effective quality assurance.
Quality control involves various activities, including test activities, that support the achievement of appropriate levels of quality. Test activities are part of the overall software development or maintenance process. Since quality assurance is concerned with the proper execution of the entire process, quality assurance supports proper testing.
==========
According to Glossary:
Quality Assurance: Part of quality management focused on providing confidence that quality requirements will be fulfilled.
Quality Control: The operational techniques and activities, part of quality management, that are focused on fulfilling quality requirements.
Quality Management: Coordinated activities to direct and control an organization with regard to quality. Direction and control with regard to quality generally includes the establishment of the quality policy and quality objectives, quality planning, quality control, quality assurance and quality improvement.
Domain
1.2 Why is Testing Necessary?

Question 37

Which of the following is not a correct statement about software testing and quality assurance ?

a) Software testing and quality assurance are not the same
b) Quality assurance supports proper testing
c) Some people use the phrase quality assurance to refer to testing
d) There must be two different roles in any type of organization, quality assurance specialist and software tester **(Correct answer)**

Overall explanation

-Software Testing & Quality Assurance are not the same: This is correct, while people often use the phrase quality assurance (or just QA) to refer to testing, quality assurance and testing are not the same, but they are related.

-Quality Assurance supports proper testing: this is correct, quality assurance is typically focused on adherence to proper processes, in order to provide confidence that the appropriate levels of quality will be achieved. When processes are carried out properly, the work products created by those processes are generally of higher quality, which contributes to defect prevention.

-Some people use the phrase quality assurance to refer to testing: people often use the phrase quality assurance to refer to testing, this is not correct but some people do it.

-There must be two different roles in any type of organization, quality assurance specialist and software tester: This is wrong, in many companies the same person works as a tester & a QA Specialist.

==========

According to Syllabus:

Quality Assurance & Testing:

While people often use the phrase quality assurance (or just QA) to refer to testing, quality assurance and testing are not the same, but they are related. A larger concept, quality management, ties them together.

Quality management includes all activities that direct and control an organization with regard to quality. Among other activities, quality management includes both quality assurance and quality control. Quality assurance is typically focused on adherence to proper processes, in order to provide confidence that the appropriate levels of quality will be achieved. When processes are carried out properly, the work products created by those processes are generally of higher quality, which contributes to defect prevention. In addition, the use of root cause analysis to detect and remove the causes of defects, along with the proper application of the findings of retrospective meetings to improve processes, are important for effective quality assurance.

Quality control involves various activities, including test activities, that support the achievement of appropriate levels of quality. Test activities are part of the overall software development or maintenance process. Since quality assurance is concerned with the proper execution of the entire process, quality assurance supports proper testing.

==========

According to Glossary:

Quality Assurance: Part of quality management focused on providing confidence that quality requirements will be fulfilled.

Quality Control: The operational techniques and activities, part of quality management, that are focused on fulfilling quality requirements.
Quality Management: Coordinated activities to direct and control an organization with regard to quality. Direction and control with regard to quality generally includes the establishment of the quality policy and quality objectives, quality planning, quality control, quality assurance and quality improvement.
Domain
1.2 Why is Testing Necessary?

Question 38
A tester was executing a test case, the test data in the test case was pronounced incorrectly. The name of item that should be written in the search field should contain the value "Laptop", but the tester wrote the value "labtob" instead.
The actual result of the test case was a message saying "There is no results for the word (labtob), did you mean (laptop)"
The tester reported the defect and written the following description "When the user searches for laptops, an error message appears, a list of laptops should appear instead"
What is the problem with this situation?

a) The tester made a mistake which is called "False Positive" **(Correct answer)**
b) The tester made a mistake which is called "False Negative"
c) The developer made a mistake in writing the code which resulted in the bug
d) There is no problem with this situation

Overall explanation
In false positive, the tester finds a defect which is not a defect. For example, the tester was testing the software and the website didn't load because the internet connection is dropped, he reported it as a defect.
In false negative, there is a defect in the software but the tester didn't find it. For example, the tester executed all the test cases for the mobile app on portrait mode but there are undiscovered defects that don't appear unless he uses the app in landscape mode, he didn't report those defects.

In the example above, the tester wrote a defect report for something that works as it is intended to word, so this is considered as a false positive.
==========
According to Syllabus:
Errors, Defects, and Failures:
Not all unexpected test results are failures. False positives may occur due to errors in the way tests were executed, or due to defects in the test data, the test environment, or other testware, or for other reasons.The inverse situation can also occur, where similar errors or defects lead to false negatives. False negatives are tests that do not detect defects that they should have detected; false positives are reported as defects, but aren’t actually defects.
==========
Domain
1.2 Why is Testing Necessary?