6 Testing - exam questions

You should know the answers to these questions

What is (a) Testing, (b) a Testing Technique, (c) a Testing Strategy

- **Testing:** The process of evaluating a system or component to determine whether it satisfies the specified requirements and to identify any defects.
- **Testing Technique:** A method or procedure used to design test cases, such as equivalence partitioning or boundary value analysis.
- **Testing Strategy:** A high-level plan that outlines the approach, resources, schedule, and scope for testing activities.

What is the difference between an error, a failure and a defect?

- **Error:** A human mistake in the design, coding, or requirements.
- **Defect:** A flaw in the system caused by an error, leading to incorrect or unexpected behavior.
- **Failure:** The manifestation of a defect when the system behaves incorrectly during execution.

What is a test case? A test stub? A test driver? A test fixture?

- **Test Case:** A set of conditions, inputs, and expected outcomes used to verify a system's behavior.
- **Test Stub:** A simulated module or component that mimics the behavior of a dependent component during testing.
- **Test Driver:** A program or script used to initiate and manage the execution of test cases.
- **Test Fixture:** A fixed state of the system under test, including pre-configured inputs, data, or environmental settings required for testing.

What are the differences and similarities between basis path testing, condition testing and loop testing?

- **Basis Path Testing:** Focuses on ensuring every independent path through the code is executed at least once.
- **Condition Testing:** Validates the correctness of logical conditions in decision statements.
- **Loop Testing:** Focuses on validating the functionality of loops, including boundary cases and nested loops.
- **Similarities:** All aim to ensure comprehensive testing and identify logical flaws.
- **Differences:** Each targets specific constructs (paths, conditions, or loops).

How many tests should you write to achieve MC/DC coverage? And multiple condition coverage?

- **MC/DC Coverage:** Requires at least one test case for each condition to independently affect the outcome.
- **Multiple Condition Coverage:** Requires test cases for all possible combinations of conditions.

Where do you situate alpha/beta testing in the four quadrants model?

- **Alpha Testing:** Quadrant 2 (Business-facing, product validation).
- **Beta Testing:** Quadrant 3 (Business-facing, customer-focused).

What are the differences and similarities between unit testing and regression testing?

- **Unit Testing:** Verifies individual components for correctness.
- **Regression Testing:** Ensures that changes or updates have not introduced new defects.
- **Similarities:** Both involve executing tests to verify behavior.
- **Differences:** Unit testing focuses on isolated components, while regression testing evaluates the impact of changes on the entire system.

How do you know when you tested enough?

Testing is considered sufficient when:

- All specified requirements are verified.
- Code coverage meets the target (e.g., 90%+ for critical systems).
- No high-severity defects remain unresolved.
- Risk levels are acceptable.
- Deadlines and budgets are met.

What is Alpha-testing and Beta-Testing? When is it used?

- **Alpha Testing:** Performed internally by developers and testers to identify bugs before releasing the product to external users.
- **Beta Testing:** Conducted by external users to gather feedback and uncover issues in a real-world environment.

What is the difference between stress-testing and performance testing?

- **Stress Testing:** Evaluates how the system behaves under extreme load or resource constraints.
- **Performance Testing:** Assesses the system's responsiveness and stability under expected load conditions.

You should be able to complete the following tasks

Complete test cases for the Loop Testing example (Loop Testing on page 19).

- Identify the loop types (simple, nested, concatenated).
- Define test cases for:
  - Zero iterations (bypassing the loop).
  - One iteration.
  - Maximum iterations.
  - Iterations just beyond the maximum (if applicable).
- Include edge cases for loop conditions and any data dependencies.

Rewrite the binary search so that basis path testing and loop testing becomes easier.

- Refactor the binary search function to:
  - Explicitly separate conditional and loop constructs.
  - Use clear, distinct paths for edge cases (e.g., element not found, first/middle/last element found).
- Simplify nested conditions and ensure each path corresponds to a unique basis path.

Write a piece of code implementing a quicksort. Apply all testing techniques (basis path testing, conditional testing [3 variants], loop testing, equivalence partitioning) to derive appropriate test cases.

- Implement quicksort using recursion and partitioning.
- Derive test cases for:
  - **Basis Path Testing:** Cover recursive calls, base cases, and partition scenarios.
  - **Condition Testing (3 Variants):**
    - Simple condition testing for pivot selection.
    - Branch condition testing for comparisons in the partition step.

        * Compound condition testing for complex conditions in loops and recursion.
    - **Loop Testing:** Validate the partition loop for zero, one, and many iterations.
    - **Equivalence Partitioning:** Partition input arrays into:
        * Sorted arrays.
        * Reverse-sorted arrays.
        * Arrays with duplicate elements.

Write FIT test cases for the user stories in your Bachelor Capstone Project.

- Identify user stories and their acceptance criteria.
- Create FIT (Framework for Integrated Testing) tables for:
    - Inputs: Parameters or actions users perform.
    - Expected Outputs: Results or behaviors the system should produce.
- Example FIT Table for a login feature: | **Input** | **Expected Output** | |——————|———————————| | Valid credentials | Login successful | | Invalid credentials | Error message displayed |

Apply fuzz testing to the REST-API of your project.

- Use fuzz testing tools (e.g., OWASP ZAP, Postman Fuzz Testing, or custom scripts) to:
    - Generate random or malformed inputs for API endpoints.
    - Test for unexpected behavior (e.g., crashes, errors, or security vulnerabilities).
- Define metrics for evaluating the results, such as error logs or response times.

Can you answer the following questions?

You're responsible for setting up a test program. To whom will you assign the responsibility to write tests? Why?

- Assign responsibilities to:
    - Developers: Write unit tests since they understand the code logic.
    - Testers/QA Engineers: Write integration and system tests to ensure objectivity.
    - Product Owners or Stakeholders: Provide input for acceptance tests to align with user expectations.

Why do we distinguish between several levels of testing in the V-model?

- To ensure thorough validation at each development stage.
- To match testing activities with corresponding phases (e.g., unit testing with implementation, system testing with integration).
- To reduce defects early and minimize costs of fixing them later.

Explain why basis path testing, condition testing, and loop testing complement each other.

- **Basis Path Testing:** Ensures all independent paths are tested.
- **Condition Testing:** Validates logical decisions and their outcomes.
- **Loop Testing:** Focuses on loop-specific scenarios.
- Combined, they address different code structures, ensuring comprehensive coverage.

Why is mutation coverage a better criterion for assessing the strength of a test suite?

- It measures the test suite's ability to detect small changes (mutations) in the code, simulating potential defects.
- A higher mutation coverage indicates a robust suite capable of catching subtle bugs.

Explain fuzzing (fuzz testing) in your own words.

- Fuzzing is a testing technique where random, invalid, or unexpected inputs are provided to a system to identify vulnerabilities, crashes, or unexpected behavior.

Explain what FIT tables are.

- FIT (Framework for Integrated Testing) tables are structured representations of test cases, showing inputs, actions, and expected outputs. They bridge communication between stakeholders and developers by aligning tests with requirements.

When would you combine top-down testing with bottom-up testing? Why?

- Combine these approaches in a **hybrid testing strategy** when:
  - Both higher-level modules (top-down) and lower-level utilities (bottom-up) are critical.
  - Immediate feedback on integration is needed while testing lower-level components.

When would you combine black-box testing with white-box testing? Why?

- Use both when:
  - You need to validate external functionality (black-box) and internal code structure (white-box).
  - The system must meet user expectations and maintain code quality.

Is it worthwhile to apply white-box testing in an OO context?

- Yes, as it verifies object interactions, method behaviors, and code-level details, ensuring robust encapsulation and inheritance implementations.

What makes regression testing important?

- Ensures that new changes or fixes do not introduce unintended defects.
- Maintains system stability over time.

Is it acceptable to deliver a system that is not 100% reliable? Why (not)?

- It depends on the context:
  - **Critical Systems (e.g., medical, aviation):** No, as reliability is essential for safety.
  - **Non-critical Systems:** Yes, if defects do not significantly impact users and deadlines must be met.

Explain the subtle difference between code coverage and test coverage.

- **Code Coverage:** Measures the percentage of code executed by tests.
- **Test Coverage:** Evaluates the extent to which requirements or functionality are tested, which may not directly correlate with code execution.