**Ans no 1.**

Black Box Testing is a software testing method in which the internal structure/design/implementation of the item being tested is not known to the tester. Only the external design and structure are tested.

White Box Testing is a software testing method in which the internal structure/design/implementation of the item being tested is known to the tester. Implementation and impact of the code are tested.

**Ans no 2.**

1. Unit Testing

Unit testing involves testing individual components or modules of a software application to ensure that each part functions correctly. This level of testing is typically performed by developers during the coding phase.

Objectives:

Verify Correctness: Ensure that each unit of the code performs as expected.

Isolate Issues: Detect and fix bugs at an early stage by isolating each part of the code.

Improve Code Quality: Enhance the overall quality and maintainability of the code.

Facilitate Refactoring: Allow developers to make changes and refactor code confidently by ensuring that unit tests will catch regressions.

2. Integration Testing

Integration testing focuses on verifying the interactions between integrated units or components. The goal is to identify issues that occur when units are combined and ensure that they work together as expected.

Objectives:

Verify Interfaces: Ensure that different modules or components communicate and interact correctly.

Detect Interaction Issues: Identify problems that occur due to integration, such as data flow issues, incorrect interfaces, and communication errors.

Validate Combined Functionality: Confirm that integrated components achieve the desired overall functionality.

3. System Testing

System testing evaluates the complete and integrated software application as a whole. It tests the system's compliance with the specified requirements to ensure that the application functions correctly in its entirety.

Objectives:

Validate Complete System: Ensure that the entire system meets the specified requirements and performs as expected.

End-to-End Testing: Test the system's end-to-end functionality, including all modules and components.

Identify System-Wide Issues: Detect issues that may not have been uncovered during unit or integration testing, such as performance, security, and usability problems.

4. User Acceptance Testing (UAT)

User Acceptance Testing (UAT) is the final level of testing conducted by the end-users or clients to ensure that the software meets their expectations and requirements. This testing validates that the software is ready for deployment.

Objectives:

Validate Requirements: Confirm that the software meets the user's business needs and requirements.

Ensure Usability: Verify that the software is user-friendly and operates as the users expect.

Acceptance Decision: Provide the basis for the client or end-users to accept the software and approve it for deployment.

**Ans no 3.**

Quality Assurance (QA) professionals play a critical role in the software development lifecycle (SDLC) by ensuring that the software meets the required quality standards before it is released to the end-users. Their responsibilities span various phases of the SDLC, from planning and design to testing and maintenance. Below are the key responsibilities of QA professionals in the software development lifecycle:

1. Requirement Analysis

Responsibilities:

Understand Requirements: QA professionals review and understand the business requirements and technical specifications to ensure clarity and completeness.

Identify Testable Requirements: Determine which requirements can be tested and ensure that they are well-defined and measurable.

Collaborate with Stakeholders: Work closely with business analysts, developers, and product owners to resolve any ambiguities in the requirements.

2. Test Planning

Responsibilities:

Develop Test Plans: Create comprehensive test plans that outline the testing strategy, scope, objectives, resources, schedule, and deliverables.

Define Test Objectives: Establish clear objectives for each testing phase (unit, integration, system, UAT).

Resource Planning: Identify the necessary resources, including tools, environments, and personnel, required for testing.

Risk Assessment: Identify potential risks and dependencies and develop mitigation strategies.

3. Test Design

Responsibilities:

Design Test Cases: Create detailed test cases and test scripts based on the requirements and test plans.

Develop Test Scenarios: Design various test scenarios, including positive and negative test cases, edge cases, and performance tests.

Test Data Preparation: Identify and prepare the necessary test data to execute the test cases.

Review Test Cases: Ensure that test cases are reviewed and approved by relevant stakeholders to ensure accuracy and completeness.

4. Test Execution

Responsibilities:

Execute Test Cases: Perform manual and automated testing as per the test plans and document the results.

Log Defects: Identify, log, and track defects using defect management tools, ensuring they are documented with sufficient detail for developers to reproduce and fix.

Retesting and Regression Testing: Conduct retesting of fixed defects and perform regression testing to ensure new changes do not affect existing functionality.

Monitor Test Progress: Track and report on the progress of testing activities, ensuring they stay on schedule.

5. Test Reporting

Responsibilities:

Document Test Results: Record and document all test results, including passed, failed, and blocked test cases.

Generate Test Reports: Produce detailed test reports that summarize the testing activities, including defect status, test coverage, and overall quality assessment.

Communicate Findings: Present test results and findings to stakeholders, providing insights into the quality of the software and any potential risks.

6. Continuous Improvement

Responsibilities:

Process Improvement: Continuously evaluate and improve testing processes, methodologies, and tools to enhance efficiency and effectiveness.

Implement Best Practices: Stay updated with industry best practices and incorporate them into the QA processes.

Feedback Loop: Provide feedback to the development and requirement teams to help improve the quality of future releases.

7. Maintenance and Support

Responsibilities:

Support Post-Release: Provide support for post-release testing and ensure that any issues reported by users are addressed promptly.

Regression Testing: Regularly perform regression testing during maintenance releases to ensure ongoing quality.

Update Test Artifacts: Keep test cases, test scripts, and documentation up to date with changes in the application.

**Ans no 5.**

import java.util.Scanner;

public class SumCalculator {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int A = scanner.nextInt();

int B = scanner.nextInt();

int C = scanner.nextInt();

int SUM = A + B + C;

System.out.println("SUM = " + SUM);

scanner.close();

}

}