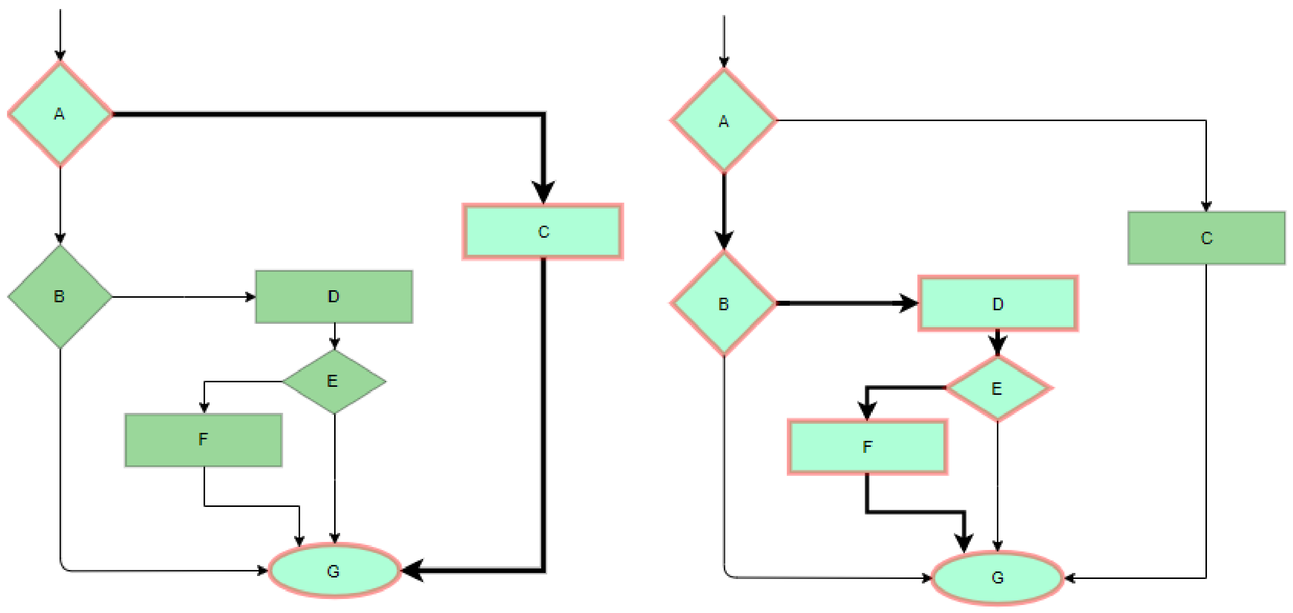
<u>What is White box testing?</u>

White box testing is a software testing method that examines the internal structure, design, and code of an application to verify its logic and identify defects

<u>White Box Testing Techniques:</u>

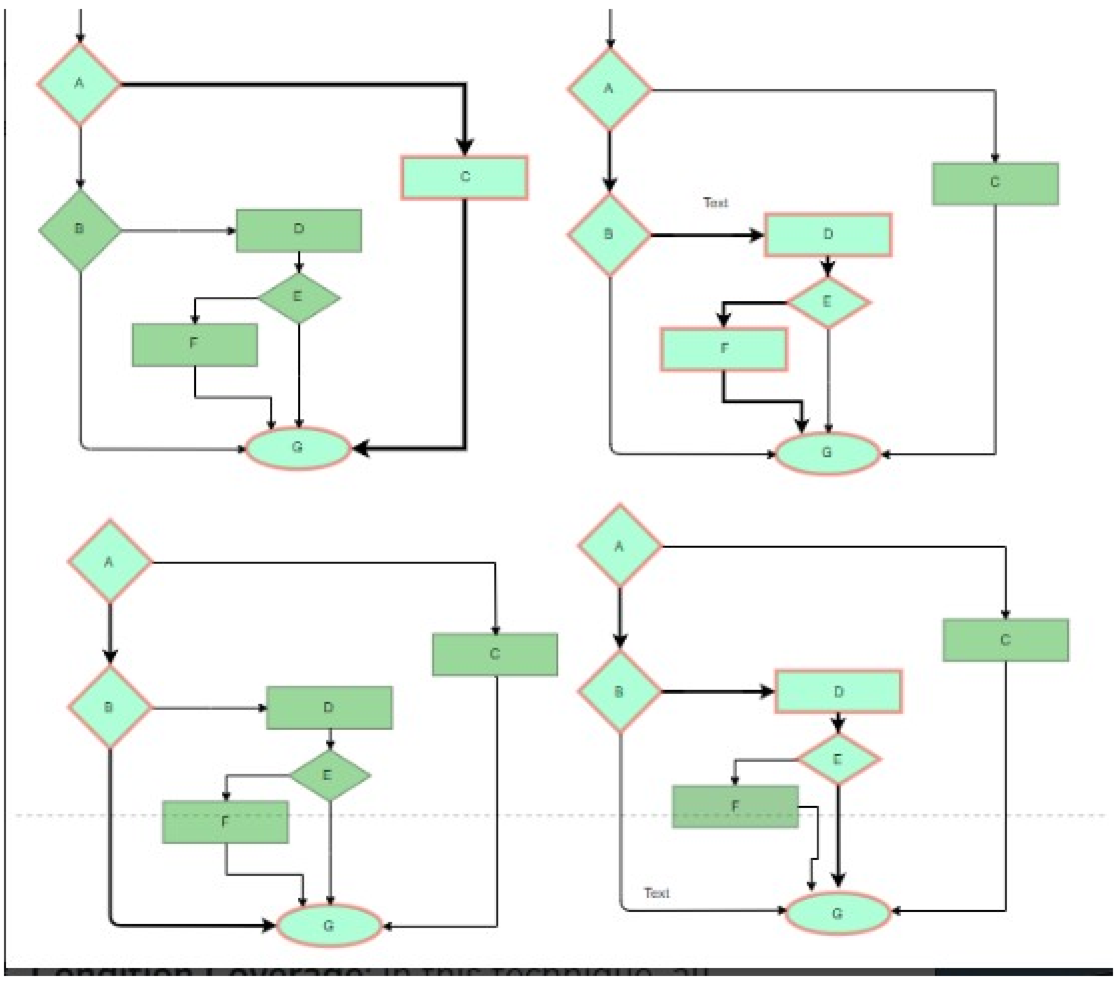
White box testing techniques include path coverage, statement coverage, and branch coverage.

1. Statement Coverage: In this technique, the aim is to traverse all statements at least once. Hence, each line of code is tested. In the case of a flowchart, every node must be traversed at least once. Since all lines of code are covered, it helps in pointing out faulty code.



2. Branch Coverage: Branch coverage focuses on testing the decision points or conditional branches in the code. It checks whether both possible outcomes (true and false) of each conditional statement are tested. In this technique, test cases are designed so that each branch from all decision points is traversed at least once. In a flowchart, all edges must be traversed at least once.

In a flowchart, all edges must be traversed at least once.



3. Condition Coverage: In this technique, all individual conditions must be covered as shown in the following example:

```
READ X, Y

IF(X == 0 || Y == 0)

PRINT '0'

#TC1 – X = 0, Y = 55

#TC2 – X = 5, Y = 0
```

4. Multiple Condition Coverage: In this technique, all the possible combinations of the possible outcomes of conditions are tested at least once. Let's consider the following example:

```
READ X, Y

IF(X == 0 || Y == 0)

PRINT '0'

#TC1: X = 0, Y = 0

#TC2: X = 0, Y = 5
```

#TC3: X = 55, Y = 0

#TC4: X = 55, Y = 5

5. Basis Path Testing: In this technique, control flow graphs are made from code or flowchart and then Cyclomatic complexity is calculated which defines the number of independent paths so that the minimal number of test cases can be designed for each independent path. Steps:

Make the corresponding control flow graph

Calculate the cyclomatic complexity

Find the independent paths

Design test cases corresponding to each independent path

V(G) = P + 1, where P is the number of predicate nodes in the flow graph

V(G) = E – N + 2, where E is the number of edges and N is the total number of nodes

V(G) = Number of non-overlapping regions in the graph

#P1: 1 – 2 – 4 – 7 – 8

#P2: 1 – 2 – 3 – 5 – 7 – 8

#P3: 1 – 2 – 3 – 6 – 7 – 8

#P4: 1 – 2 – 4 – 7 – 1 – . . . – 7 – 8

6. Loop Testing: Loops are widely used and these are fundamental to many algorithms hence, their testing is very important. Errors often occur at the beginnings and ends of loops.

Simple loops: For simple loops of size n, test cases are designed that:

Skip the loop entirely

Only one pass through the loop

2 passes

m passes, where m < n

n-1 ans n+1 passes

Nested loops: For nested loops, all the loops are set to their minimum count, and we start from the innermost loop. Simple loop tests are conducted for the innermost loop and this is worked outwards till all the loops have been tested.

Concatenated loops: Independent loops, one after another. Simple loop tests are applied for each. If they’re not independent, treat them like nesting.

Key points to consider:

Internal focus: It tests the internal structure and logic of the software, rather than just the external functionality from a user's perspective.

Developer-centric: It is often performed by developers who have an in-depth understanding of the codebase.

Goal: To find flaws in the code and verify that internal operations are performed correctly, improve security, and ensure the quality of the software's architecture.

Methods: Common techniques include statement coverage (ensuring every line of code is executed), branch coverage (ensuring every branch of a conditional statement is executed), and path coverage.

Automation: It can be highly automated as it relies on the internal structure, but it also requires significant maintenance when the code is updated.

Names: It is also known as clear box, glass box, or open box testing.

White Box Testing Focus on Following things:

1. Code Logic and Flow

Checks if the program's logic works as intended. This means verifying that code modules (like functions or classes) interact correctly and that control structures such as if-else statements, loops, or switches execute properly. For example, ensuring a login function redirects users correctly based on valid or invalid credentials.

2. Code Coverage

Ensures tests exercise as much of the code as possible. This includes:

Statement coverage: Every line of code runs at least once.

Branch coverage: All decision paths (e.g., true/false conditions) are tested.

Path coverage: Every possible route through the code is checked.

This helps find untested or "dead" code that could hide bugs.

3. Data Flow and Variables

Verifies that data is passed and manipulated correctly through the application. This includes ensuring variables are properly initialized, updated, and used without causing any errors or unexpected behavior. Monitoring the flow of data ensures that the system remains stable and reliable as it processes inputs, calculations, and outputs.

4. Internal Functions and Methods

Tests the individual functions or methods to ensure they perform their intended tasks accurately and return the expected results. This part of white-box testing focuses on validating business logic, mathematical computations, and other operations within the software. Ensuring these internal processes are correct helps catch potential issues early and improves the reliability of the overall system.

5. Boundary Conditions

Examines how the code handles edge cases, like the maximum or minimum values for inputs (e.g., a loop running 0 or 100 times, or an input field accepting a 255-character string). This ensures the app doesn't crash at its limits.

6. Error Handling and Exception Management

Confirms the program manages errors smoothly, catching exceptions (e.g., invalid inputs) and providing clear feedback. For example, testing if a file upload function handles a missing file gracefully.

Tools of White box testing:

SonarQube

Veracode

OWASP Code Pulse

JaCoCo

PVS-Studio

Checkmarx

Coverity

Klocwork

CodeClimate

Codacy

These tools help developers to perform detailed analysis on the source code, checking that all potential issues are detected and addressed early in the development cycle.

Advantages of White Box Testing

Here are the Advantages of White Box Testing:

Thorough Testing: White box testing is thorough as the entire code and structures are tested.

Code Optimization: It results in the optimization of code removing errors and helps in removing extra lines of code.

Early Detection of Defects: It can start at an earlier stage as it doesn't require any interface as in the case of black box testing.

Integration with SDLC: White box testing can be easily started in Software Development Life Cycle.

Detection of Complex Defects: Testers can identify defects that cannot be detected through other testing techniques.

Comprehensive Test Cases: Testers can create more comprehensive and effective test cases that cover all code paths.

Disadvantages of White Box Testing

Here are the Disadvantages of White Box Testing:

Programming Knowledge and Source Code Access: Testers need to have programming knowledge and access to the source code to perform tests.

Overemphasis on Internal Workings: Testers may focus too much on the internal workings of the software and may miss external issues.

Bias in Testing: Testers may have a biased view of the software since they are familiar with its internal workings.

Test Case Overhead: Redesigning code and rewriting code needs test cases to be written again.

Dependency on Tester Expertise: Testers are required to have in-depth knowledge of the code and programming language as opposed to black-box testing.

Inability to Detect Missing Functionalities: Missing functionalities cannot be detected as the code that exists is tested.