Testing

Difference between Black Box Testing and Gray Box Testing

Black Box Testing: Black Box Testing is a Software Testing technique in which the tester doesn't know the internal structure, design and implementation of the software application that is being tested.

Gray Box Testing: Gray Box Testing is a software testing technique which is a combination of Black Box Testing technique and White Box Testing technique. The internal structure, design and implementation is partially known in Gray Box Testing.

Differences between Black Box Testing and Gray Box Testing:

Black Box Testing	Gray Box Testing
It is a software testing technique in which the tester doesn't know the internal structure of the application being tested.	It is a software testing technique in which the tester partially know the internal structure of the application being tested.
It is known as closed box testing.	It is known as translucent testing.
No knowledge of implementation is required.	Knowledge of implementation is required but need not to be expert.
It is based on external expectations and outer behavior of the software.	It is based on database diagrams and data flow diagrams.
It is less time consuming.	It is time consuming but not too much.
It is done by trial and error method.	It is done on the basis of data domains.
It improves some of the qualities of the software.	It improves overall quality of the software.

Differences between White Box Testing and Gray Box Testing

White Box Testing: White Box Testing is a type of Software Testing in which the internal structure, design and implementation of the software application that is being tested is fully known to the tester.

Gray Box Testing: Gray Box Testing is a software testing technique which is a combination of Black Box Testing technique and White Box Testing technique. The internal structure, design and implementation is partially known in Gray Box Testing.

Differences between White Box Testing and Gray Box Testing:

WHITE BOX TESTING	GRAY BOX TESTING
It is a type of software testing in which the internal structure and design of the software application is fully known to the tester.	It is a type of software testing in which the internal structure and design of the software application is partially known to the tester.
It is also known as clear box testing or transparent testing.	It is known as translucent testing.
It is performed by testers and developers.	It is performed by end users, testers and developers.

WHITE BOX TESTING	GRAY BOX TESTING
Full knowledge of the implementation is required.	Small knowledge of the implementation is enough.
High programming skills are required to perform white box testing.	basic programming skills are enough to perform this testing.
It is a time consuming testing.	It is a less time consuming testing.
It is used for algorithm testing.	It is not used in algorithm testing.

Black box testing–

Black box testing is a type of software testing in which the functionality of the software is not known. The testing is done without the internal knowledge of the products.

Black box testing can be done in following ways:

1. Syntax Driven Testing – This type of testing is applied to systems that can be syntactically represented by some language. For example- compilers, language that can be represented by context free grammar. In this, the test cases are generated so that each grammar rule is used at least once.

2. Equivalence partitioning – It is often seen that many type of inputs work similarly so instead of giving all of them separately we can group them together and test only one input of each group. The idea is to partition the input domain of the system into a number of equivalence classes such that each member of class works in a similar way, i.e., if a test case in one class results in some error, other members of class would also result into same error.

The technique involves two steps:

- Identification of equivalence class – Partition any input domain into minimum two sets: valid values and invalid values. For example, if the valid range is 0 to 100 then select one valid input like 49 and one invalid like 104.
- Generating test cases –

  (i) To each valid and invalid class of input assign unique identification number.
  (ii) Write test case covering all valid and invalid test case considering that no two invalid inputs mask each other.

  To calculate the square root of a number, the equivalence classes will be:
  (a) Valid inputs:

  - Whole number which is a perfect square- output will be an integer.
  - Whole number which is not a perfect square- output will be decimal number.
  - Positive decimals

  (b) Invalid inputs:

  - Negative numbers(integer or decimal).
  - Characters other that numbers like “a”,”!”,”;”,etc.

3. Boundary value analysis –Boundaries are very good places for errors to occur. Hence if test cases are designed for boundary values of input domain then the efficiency of testing improves and probability of finding errors also increase. For example – If valid range is 10 to 100 then test for 10,100 also apart from valid and invalid inputs.

4. Cause effect Graphing – This technique establishes relationship between logical input called causes with corresponding actions called effect. The causes and effects are represented using Boolean graphs. The following steps are followed:

- Identify inputs (causes) and outputs (effect).
- Develop cause effect graph.
- Transform the graph into decision table.
- Convert decision table rules to test cases.

For example, in the following cause effect graph:

It can be converted into decision table like:

Each column corresponds to a rule which will become a test case for testing. So there will be 4 test cases.

5. Requirement based testing – It includes validating the requirements given in SRS of software system.

6. Compatibility testing – The test case result not only depend on product but also infrastructure for delivering functionality. When the infrastructure parameters are changed it is still expected to work properly. Some parameters that generally affect compatibility of software are:

- Processor (Pentium 3,Pentium 4) and number of processors.
- Architecture and characteristic of machine (32 bit or 64 bit).
- Back-end components such as database servers.
- Operating System (Windows, Linux, etc).

Advantages / Pros of Black Box Testing

- Unbiased tests because the designer and tester work independently
- Tester is free from any pressure of knowledge of specific programming languages to test the reliability and functionality of an application / software
- Facilitates identification of contradictions and vagueness in functional specifications
- Test is performed from a user's point-of-view and not of the designer's
- Test cases can be designed immediately after the completion of specifications

Disadvantages / Cons of Black Box Testing

- Tests can be redundant if already run by the software designer
- Test cases are extremely difficult to be designed without clear and concise specifications

- Testing every possible input stream is not possible because it is time-consuming and this would eventually leave many program paths untested
- Results might be overestimated at times
- Cannot be used for testing complex segments of code

White box Testing–

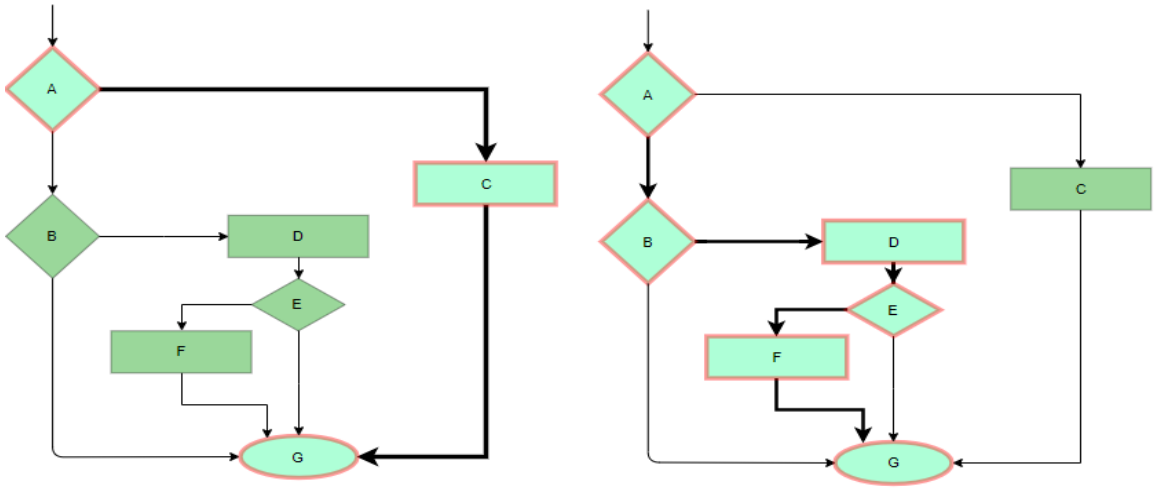
White box testing techniques analyse the internal structures the used data structures, internal design, code structure and the working of the software rather than just the functionality as in black box testing. It is also called glass box testing or clear box testing or structural testing.

Working process of white box testing:

- **Input:** Requirements, Functional specifications, design documents, source code.
- **Processing:** Performing risk analysis for guiding through the entire process.
- **Proper test planning:** Designing test cases so as to cover entire code. Execute rinse-repeat until error-free software is reached. Also, the results are communicated.
- **Output:** Preparing final report of the entire testing process.

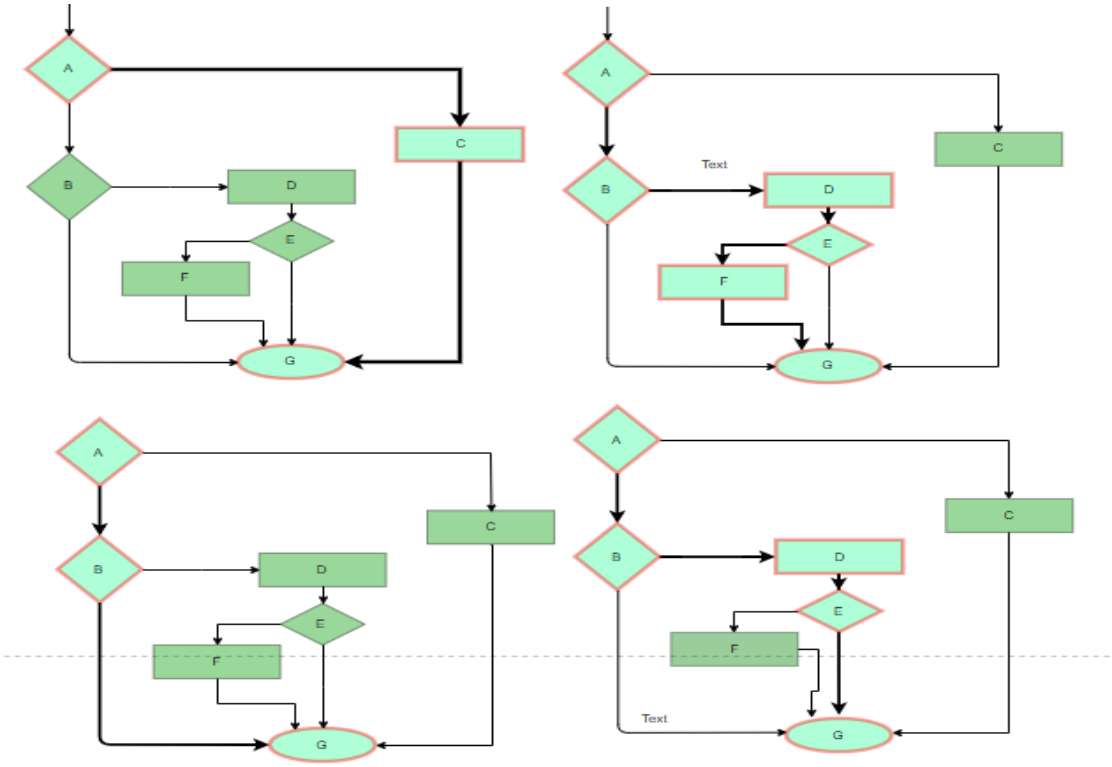
Testing techniques:

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



Statement Coverage Example

- **Branch Coverge:** In this technique, test cases are designed so that each branch from all decision points are traversed at least once. In a flowchart, all edges must be traversed at least once.



4 test cases required such that all branches of all decisions are covered, i.e, all edges of flowchart are covered

---

- **Condition Coverage:** In this technique, all individual conditions must be covered as shown in the following example:
  1. READ X, Y
  2. IF(X == 0 || Y == 0)
  3. PRINT '0'

  In this example, there are 2 conditions: X == 0 and Y == 0. Now, test these conditions get TRUE and FALSE as their values. One possible example would be:

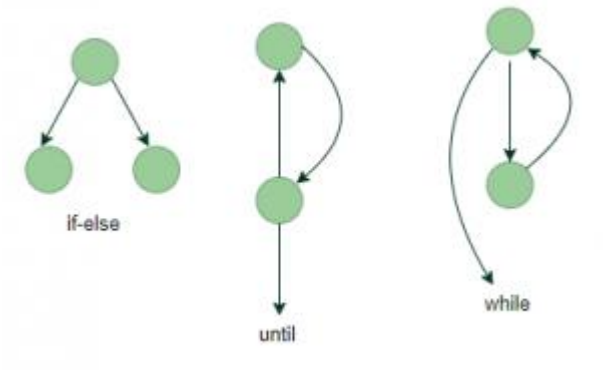
  - #TC1 – X = 0, Y = 55
  - #TC2 – X = 5, Y = 0
- **Multiple Condition Coverage:** In this technique, all the possible combinations of the possible outcomes of conditions are tested at least once. Let's consider the following example:
  0. READ X, Y

1. IF(X == 0 || Y == 0)
2. PRINT ‘0’
   - #TC1: X = 0, Y = 0
   - #TC2: X = 0, Y = 5
   - #TC3: X = 55, Y = 0
   - #TC4: X = 55, Y = 5

Hence, four test cases required for two individual conditions.
Similarly, if there are n conditions then 2^n test cases would be required.

- **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:
  0. Make the corresponding control flow graph
  1. Calculate the cyclomatic complexity
  2. Find the independent paths
  3. Design test cases corresponding to each independent path

  Flow graph notation: It is a directed graph consisting of nodes and edges. Each node represents a sequence of statements, or a decision point. A predicate node is the one that represents a decision point that contains a condition after which the graph splits. Regions are bounded by nodes and edges.

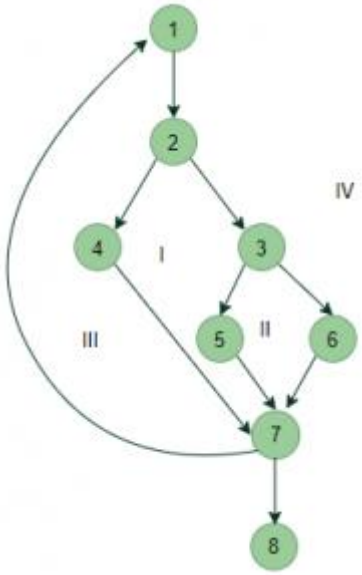


Cyclomatic Complexity: It is a measure of the logical complexity of the software and is used to define the number of independent paths. For a graph G, V(G) is its cyclomatic complexity.
Calculating V(G):

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

5. V(G) = E – N + 2, where E is the number of edges and N is the total number of nodes
6. V(G) = Number of non-overlapping regions in the graph

Example:



V(G) = 4 (Using any of the above formulae)
No of independent paths = 4

- #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

- **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.
  0. **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
  1. **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.

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

Advantages:

1. White box testing is very thorough as the entire code and structures are tested.
2. It results in the optimization of code removing error and helps in removing extra lines of code.
3. It can start at an earlier stage as it doesn't require any interface as in case of black box testing.
4. Easy to automate.

Disadvantages:

1. Main disadvantage is that it is very expensive.
2. Redesign of code and rewriting code needs test cases to be written again.
3. Testers are required to have in-depth knowledge of the code and programming language as opposed to black box testing.
4. Missing functionalities cannot be detected as the code that exists is tested.
5. Very complex and at times not realistic.

Question: a) Difference between white box testing vs Black box testing.

b) what are the advantage and disadvantage of gray box testing?