White Box Testing

What is White Box Testing?

If we go by the definition, "White box testing" (also known as clear, glass box or structural testing) is a testing technique which evaluates the code and the internal structure of a program.

White box testing involves looking at the structure of the code. When you know the internal structure of a product, tests can be conducted to ensure that the internal operations performed according to the specification. And all internal components have been adequately exercised.

Steps to Perform WBT

Step #1 – Understand the functionality of an application through its source code. Which means that a tester must be well versed with the programming language and the other tools as well techniques used to develop the software.

Step #2– Create the tests and execute them.

3 Main White Box Testing Techniques:

1. Statement Coverage
2. Branch Coverage
3. Path Coverage

Note that the statement, branch or path coverage does not identify any bug or defect that needs to be fixed. It only identifies those lines of code which are either never executed or remains untouched. Based on this further testing can be focused on.

Let's understand these techniques one by one with a simple example.

#1) Statement coverage:

In a programming language, a statement is nothing but the line of code or instruction for the computer to understand and act accordingly. A statement becomes an executable statement when it gets compiled and converted into the object code and performs the action when the program is in a running mode.

Hence *"Statement Coverage"*, as the name itself suggests, it is the method of validating whether each and every line of the code is executed at least once.

#2) Branch Coverage:

"Branch" in a programming language is like the "IF statements". An IF statement has two branches: T**rue and False**.

So, in Branch coverage (also called Decision coverage), we validate whether each branch is executed at least once.

In case of an "IF statement", there will be two test conditions:

- One to validate the true branch and,
- Other to validate the false branch.

Hence, in theory, Branch Coverage is a testing method which is when executed ensures that each and every branch from each decision point is executed.

#3) Path Coverage

Path coverage tests all the paths of the program. This is a comprehensive technique which ensures that all the paths of the program are traversed at least once. Path Coverage is even more powerful than Branch coverage. This technique is useful for testing the complex programs.

White Box Testing Example

Consider the below simple pseudocode:

```
INPUT A & B
C = A + B
IF C>100
PRINT "ITS DONE"
```

For ***Statement Coverage*** – we would only need one test case to check all the lines of the code.

That means:

If I consider *TestCase_01 to be (A=40 and B=70),* then all the lines of code will be executed.

Now the question arises:

1. Is that sufficient?
2. What if I consider my Test case as A=33 and B=45?

Because Statement coverage will only cover the true side, for the pseudo code, only one test case would NOT be sufficient to test it. As a tester, we have to consider the negative cases as well.

Hence for maximum coverage, we need to consider **"*Branch Coverage*"**, which will evaluate the "FALSE" conditions.

In the real world, you may add appropriate statements when the condition fails.

So now the pseudocode becomes:

```
INPUT A & B

C = A + B
IF C>100
PRINT "ITS DONE"
ELSE

PRINT "ITS PENDING"
```

Since Statement coverage is not sufficient to test the entire pseudo code, we would require Branch coverage to ensure maximum coverage**.**

So, for Branch coverage, we would require two test cases to complete the testing of this pseudo code.

TestCase_01: A=50, B=60

TestCase_02: A=25, B=30

With this, we can see that each and every line of the code is executed at least once.

Here are the Conclusions that are derived so far:

- Branch Coverage ensures more coverage than Statement coverage.

- Branch coverage is more powerful than Statement coverage.
- 100% Branch coverage itself means 100% statement coverage.
- But 100 % statement coverage does not guarantee 100% branch coverage.
- Now let's move on to ***Path Coverage:***

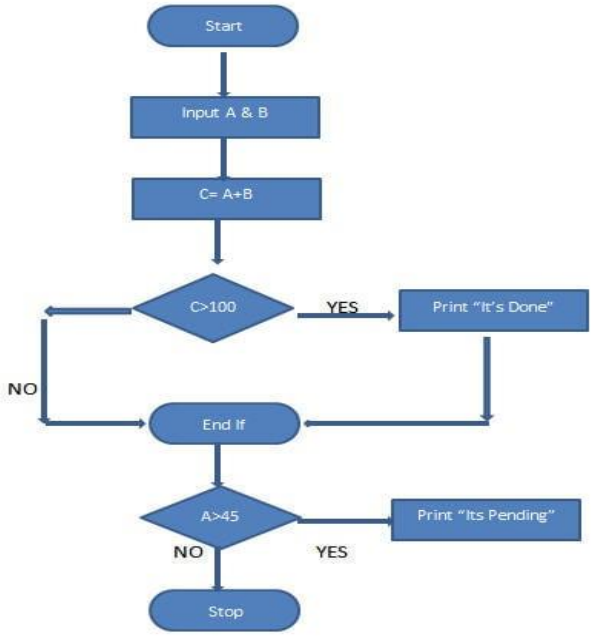
Consider this pseudocode:

```
INPUT A & B
C = A + B
IF C>100
PRINT "ITS DONE"

END IF
IF A>50
PRINT "ITS PENDING"
END IF
```

Now to ensure maximum coverage, we would require 4 test cases.

How? Simply – there are 2 decision statements, so for each decision statement, we would need two branches to test. One for true and the other for the false condition. So for 2 decision statements, we would require 2 test cases to test the true side and 2 test cases to test the false side, which makes a total of 4 test cases.



Path Coverage

In order to have the full coverage, we would need following test cases:

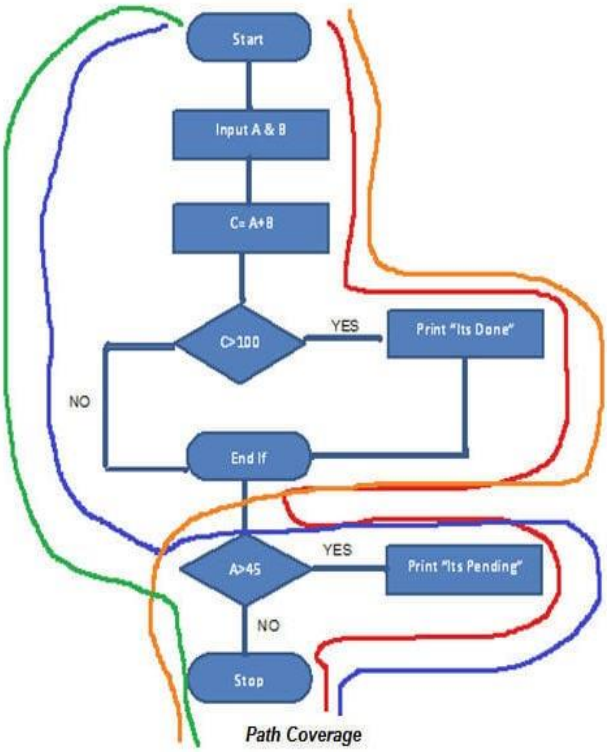
TestCase_01: A=50, B=60

TestCase_02: A=55, B=40

TestCase_03: A=40, B=65

TestCase_04: A=30, B=30

So, the path covered will be:



Red Line – TestCase_01 = (A=50, B=60)

Blue Line = TestCase_02 = (A=55, B=40)

Orange Line = TestCase_03 = (A=40, B=65)

Green Line = TestCase_04 = (A=30, B=30

Muntasir Abdullah Mizan
muntasir.abdullah01@gmail.com