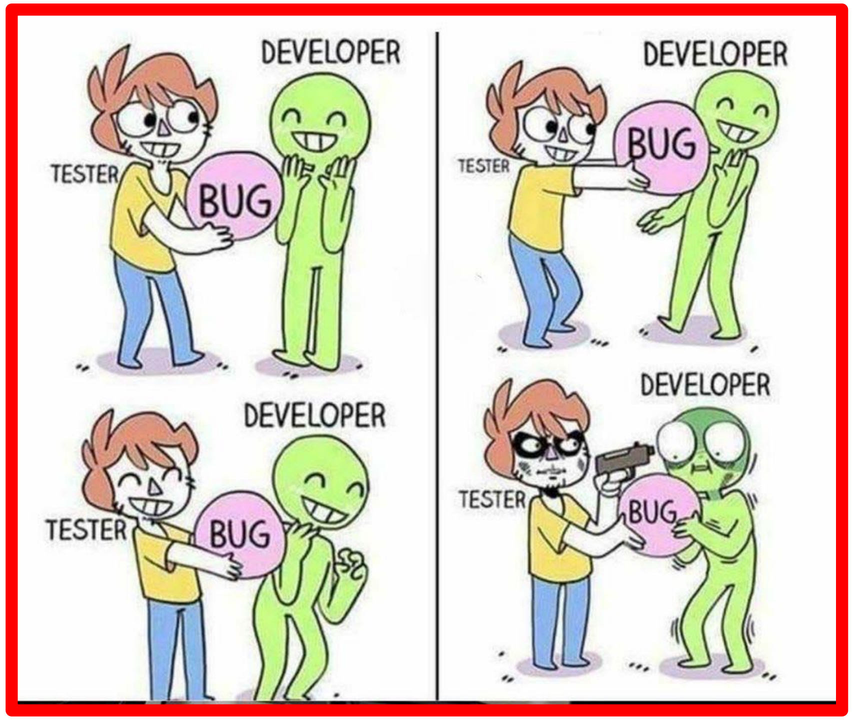


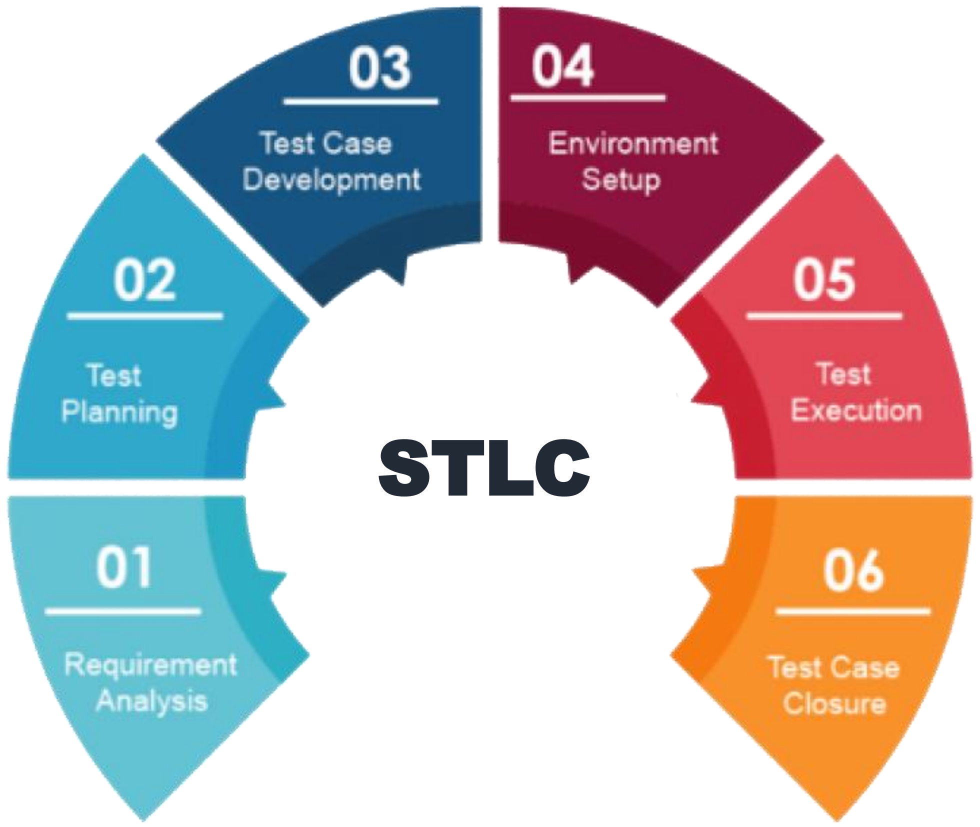
STLC

Software Testing Life Cycle

STLC is a sequence of different activities performed by the testing team to ensure the quality of the software or the product. STLC is an integral part of Software Development Life Cycle (SDLC). But, STLC deals only with the testing phases.

Manual Testing

Manual testing is a software testing process in which test cases are executed manually without using any automated tool.



Requirement Analysis: In this phase quality assurance team understands the requirements like what is to be tested.

Test Planning: Test Planning is most efficient phase of software testing life cycle where all testing plans are defined. In this phase manager of the testing team calculates estimated effort and cost for the testing work

Test Case Development: The test case development phase gets started once the test planning phase is completed. In this phase testing team note down the detailed test cases.

Environment Setup: Basically test environment decides the conditions on which software is tested. This is independent activity and can be started along with test case development.

Test Execution: After the test case development and test environment setup test execution phase gets started. In this phase testing team start executing test cases based on prepared test cases in the earlier step.

Test Case Closure: This is the last stage of STLC in which the process of testing is analyzed.

Seven Principles of Testing

01

Testing shows the presence of defects

02

Exhaustive testing is not possible

03

Early testing

04

Defect clustering

05

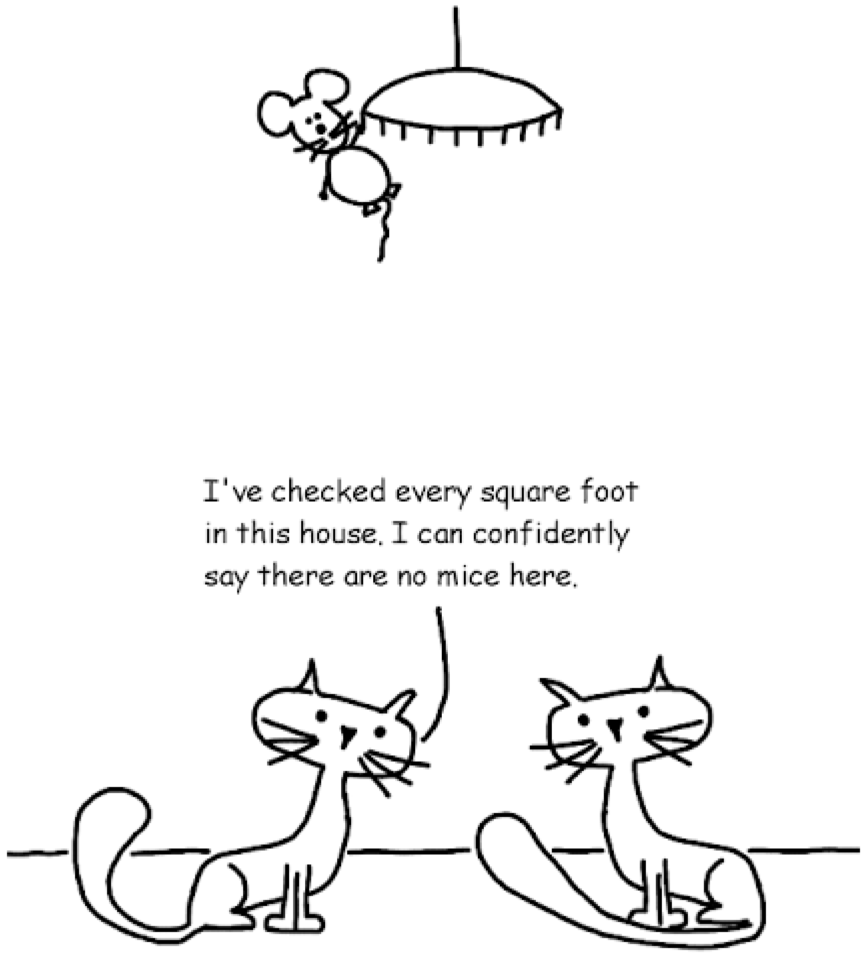
Pesticide paradox

06

Testing is context-dependent

07

Absence of errors fallacy


I've checked every square foot in this house. I can confidently say there are no mice here.

REMEMBER

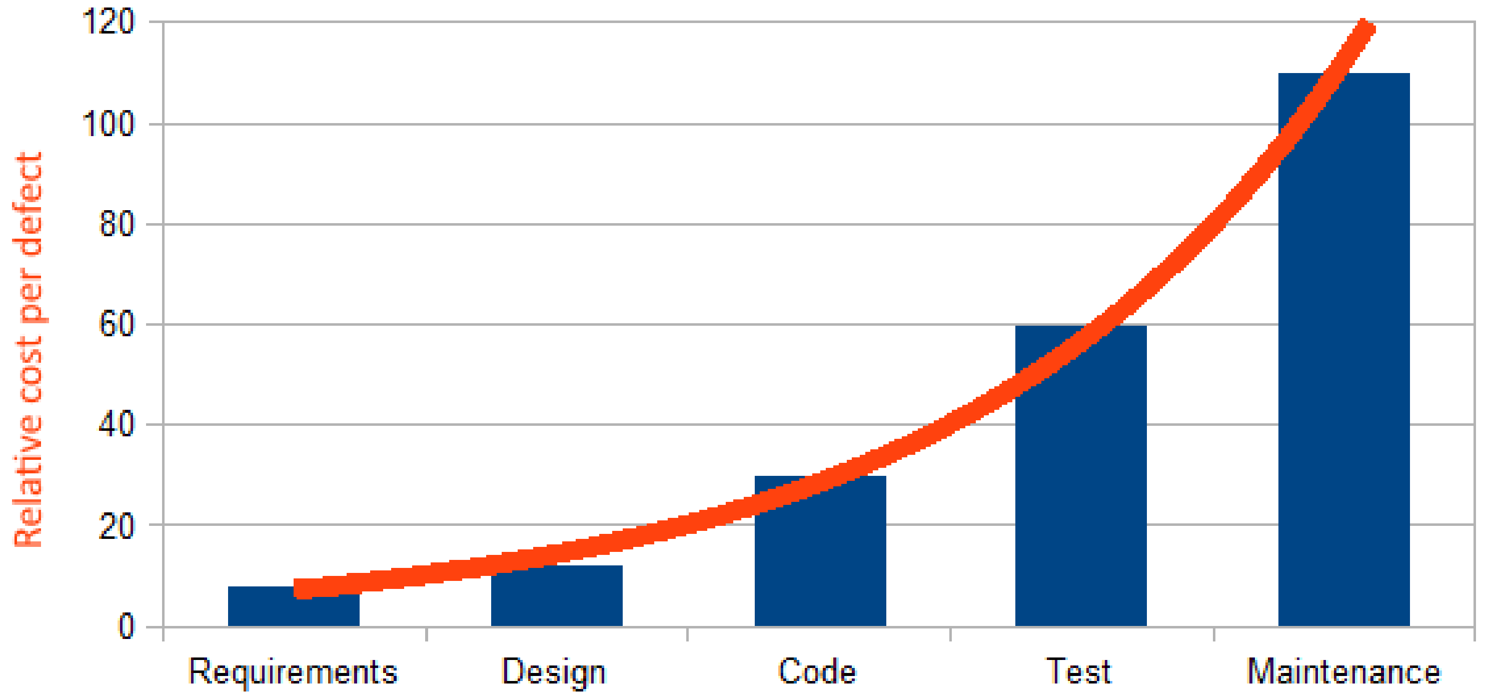
"Absence of proof is not proof of absence"

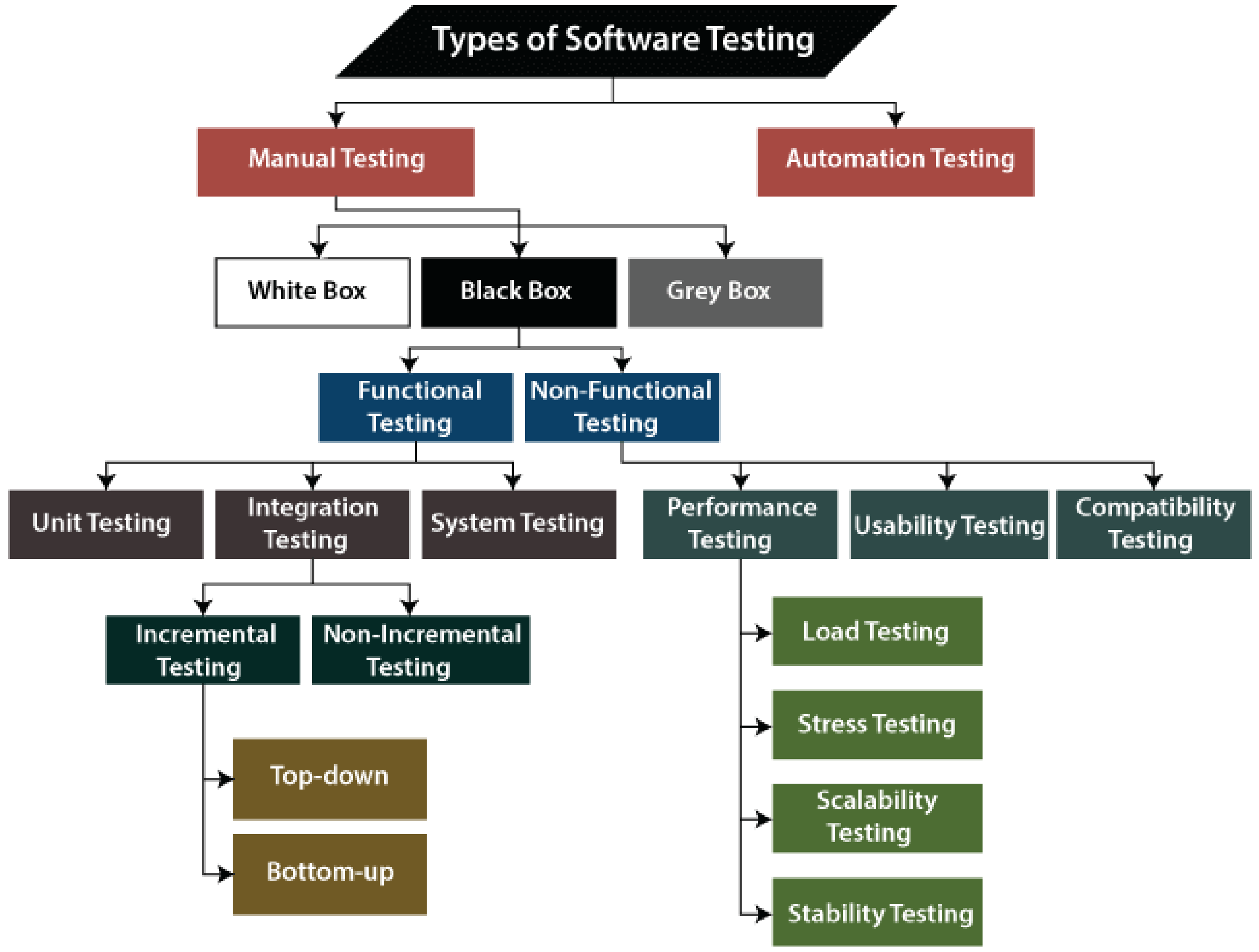
COST of Defects

DETAILS

The cost of defects identified during Software Testing, completely depends on the impact of the defects found. The earlier the defect is found, easier and less costly it is to fix these defects.

For instance, if there is a defect found in the project requirement specifications and analysis, then it is relatively cheaper to fix it. Similarly, if the defects or failures are found in the design of the software, then the product design is corrected and then re-issued. However, if these defects somehow get missed by testers and if they are identified during the user acceptance phase, then it can be way too expensive to fix such type of errors.



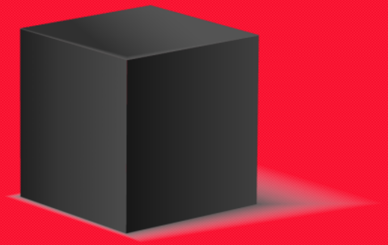

Types of Software Testing
Manual Testing
Automation Testing
White Box
Black Box
Grey Box
Functional Testing
Non-Functional Testing
Unit Testing
Integration Testing
System Testing
Performance Testing
Usability Testing
Compatibility Testing
Incremental Testing
Non-Incremental Testing
Top-down
Bottom-up
Load Testing
Stress Testing
Scalability Testing
Stability Testing

Static Testing: Before doing the white box testing done by the developer what all activities we do that comes under static testing. It is a part of Verification activity .Here we will review the code ,review the requirement, well will review the customer requirement specification, software requirement specification, high level design, low level design.

Dynamic Testing: Whenever a developer is giving the build to the test engineers, after giving the build what test engineers will do it is called Dynamic testing.It is a part of validation activity. Here we do component testing, system testing, accepting testing in one word all our functional and non-functional testing.

BLACK BOX TESTING

- Equivalence Partitioning
- Boundary Value Analysis
- State Transition Testing
- Decision Tables
- Use Case based testing



Black box testing is nothing but testing the functionality of an application without actually looking at the code that is called as black box testing.

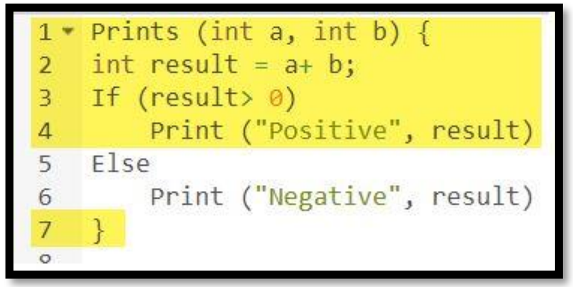
WHITE BOX TESTING

- Statement Coverage
- Condition Coverage
- Branch Coverage
- Path Coverage

White box testing is nothing but testing the internal structure, design and coding of software which is called as white box testing.

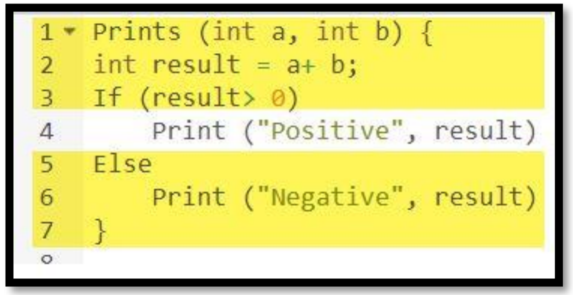
Statement Coverage

Scenario-1:: a=3,b=9



```
Prints (int a, int b) {
int result = a+ b;
If (result> 0)
    Print ("Positive", result)
Else
    Print ("Negative", result)
}
```

Scenario-1:: a=-3,b=-9



```
Prints (int a, int b) {
int result = a+ b;
If (result> 0)
    Print ("Positive", result)
Else
    Print ("Negative", result)
}
```

Statement Coverage is a white box testing technique in which all the executable statements in the source code are executed at least once. It is used for calculation of the number of statements in source code which have been executed. The main purpose of Statement Coverage is to cover all the possible paths, lines and statements in source code.

- $$test\ coverage = \frac{lines\ of\ code\ covered\ by\ tests}{total\ lines\ of\ code} * 100$$

Branches:

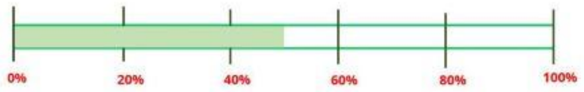
- 2,5,6,7
- 2,3,4,7

To cover all the branches we would require 2 test cases:

Test case #1 (A = 12)

Branch coverage = (Total branch covered/Total Branches)* 100

=(1/2)*100

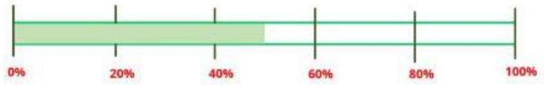


In the above code, 50% branch coverage is achieved by test case #1.

Test case #2 (A = 10)

Branch coverage = (Total branch covered/Total Branches)* 100

=(1/2)*100

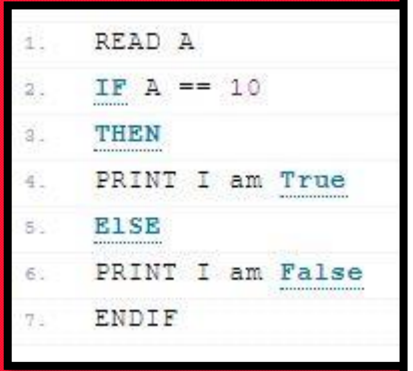


Again, 50% branch coverage is achieved by test case #2.

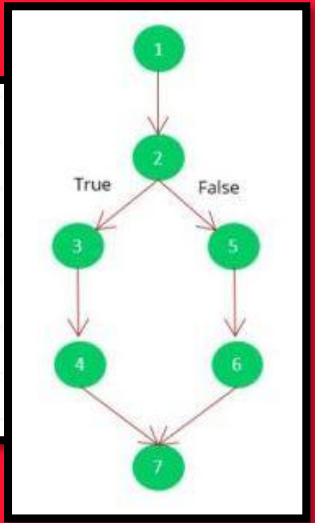
Altogether these two test cases executed all the possible branches with 50% branch coverage each.

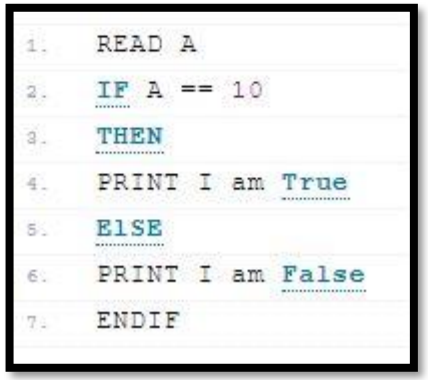
Branch Coverage

Branch coverage technique is used to cover all branches of the control flow graph. It covers all the possible outcomes (true and false) of each condition of decision point at least once. Branch coverage technique is a whitebox testing technique that ensures that every branch of each decision point must be executed.

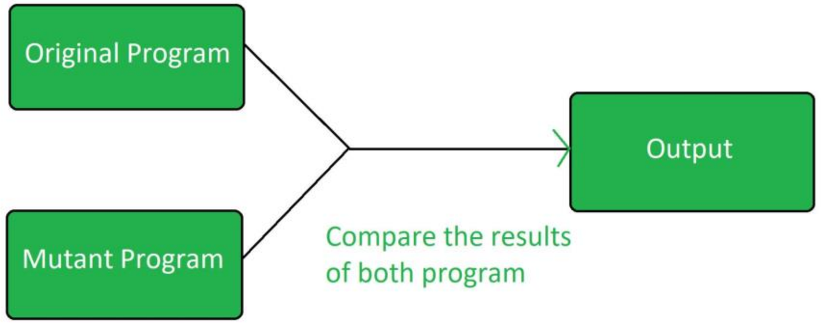


```
1.  READ A
2.  IF A == 10
3.  THEN
4.  PRINT I am True
5.  ElSE
6.  PRINT I am False
7.  ENDIF
```





```
READ A
IF A == 10
THEN
PRINT I am True
ElSE
PRINT I am False
ENDIF
```



Mutation Testing

Mutation testing, also known as code mutation testing or mutation analysis, is a form of testing in which we change specific components of an application’s source code and check whether our tests will fail as a result. We expect them to fail, as it means that the test suite catches the bugs when you expect it to. In other words, if you change your source code and your tests don’t fail, you didn’t write good tests.

GREY BOX TESTING

Grey Box Testing is a software testing technique to test a software product or application with partial knowledge of the internal structure of the application.