ASSIGNMENT NO: 1

TITLE: Black Box Functional Testing Using Manual Testing Techniques

AIM: To design and develop black box functional test cases using manual testing techniques for the given software modules.

OBJECTIVES:

1. To understand the concept of black box testing.
2. To apply Equivalence Class Partitioning and Boundary Value Analysis techniques.
3. To design test cases using Cause-and-Effect Graphing.
4. To understand State Transition Testing.
5. To perform manual test case execution using Klaros Test Management tool.

THEORY

1. Introduction to Software Testing

Software testing is the process of evaluating a software system to ensure that it satisfies specified requirements and works correctly under defined conditions. The main objective of testing is to detect defects and ensure quality.
Testing can be broadly classified into:

- White Box Testing
- Black Box Testing

In this experiment, the focus is on **Black Box Testing**.

2. Black Box Testing

Black Box Testing is a software testing technique in which I test the system without knowing its internal code, logic, or implementation details. The system is treated as a "black box," where only inputs and outputs are visible.
In this approach:

- I provide input to the system.
- I observe the output.
- I verify whether the functionality meets the specification.

The internal program structure is not considered.

Features of Black Box Testing:

- Based on functional requirements
- No knowledge of programming required
- Focuses on user perspective
- Validates system behavior

Advantages:

- Suitable for large systems
- Independent of code structure
- Effective for requirement validation

Limitations:

- Limited internal path coverage
- Difficult to identify exact root cause of defect

3. Input Domain and Test Case Selection

Every module has an **input domain**, which includes all possible valid and invalid inputs.
Since testing all possible inputs (exhaustive testing) is not practical, I use systematic techniques to select a suitable subset of test inputs that provide maximum coverage.
The techniques used in black box testing include:

1. Random Testing
2. Equivalence Class Partitioning
3. Boundary Value Analysis
4. Cause and Effect Graphing
5. State Transition Testing
6. Error Guessing

BLACK BOX TESTING TECHNIQUES

4. Random Testing

Random testing involves selecting test inputs randomly from the input domain without any structured strategy.
Example:
If the valid input range is 1 to 100, I may choose 55, 10, or 87 randomly.
Although simple, this method:

- Does not guarantee boundary coverage.
- May miss critical edge cases.

5. Equivalence Class Partitioning (ECP)

Definition:

Equivalence Class Partitioning is a technique in which the input domain is divided into groups (equivalence classes) such that all values in one class are expected to produce similar behavior.

Types of Equivalence Classes:

1. Valid Equivalence Class
2. Invalid Equivalence Class

Instead of testing all possible inputs, I select one representative value from each class.

Purpose:

- Reduce number of test cases
- Avoid redundant testing
- Ensure broad input coverage

Advantages:

- Efficient test design
- Saves time
- Covers large domain with fewer cases

Limitation:

- Does not handle combinations of conditions effectively

6. Boundary Value Analysis (BVA)

Definition:

Boundary Value Analysis is a technique that focuses on testing values at the edges (boundaries) of input ranges.Most defects occur at boundary conditions; therefore, testing boundary values increases defect detection probability.

Boundary Terms:

- BLB – Below Lower Bound
- LB – Lower Bound
- ALB – Above Lower Bound
- BUB – Below Upper Bound
- UB – Upper Bound
- AUB – Above Upper Bound

Example:

If input range is 3–15:

- BLB = 2
- LB = 3
- ALB = 4
- BUB = 14
- UB = 15
- AUB = 16

Advantages:

- Detects off-by-one errors
- Highly effective for numeric ranges

Limitation:

- Applicable only when ordered data exists

7. Cause and Effect Graphing

Definition:

Cause and Effect Graphing is a technique used to identify logical relationships between input conditions (causes) and output conditions (effects).

- Cause → Input condition
- Effect → Output condition

Logical operators such as AND, OR, and NOT are used to connect causes and effects.

Steps Involved:

1. Identify causes and effects.
2. Draw a cause-effect graph.
3. Convert graph into decision table.
4. Derive test cases from the decision table.

Advantages:

- Handles multiple input combinations
- Detects logical inconsistencies

Limitation:

- Time-consuming for complex systems

8. State Transition Testing

Definition:

State Transition Testing is used when system behavior depends on previous states.

A system is modeled as a Finite State Machine (FSM) consisting of:

- States
- Transitions
- Events

Example States:

- Empty
- Normal
- Full
- Error

Advantages:

- Tests state-based behavior
- Detects invalid transitions

Limitation:

- Difficult when number of states is large

9. Error Guessing
Definition:
Error Guessing is an experience-based testing technique in which I predict possible defect areas.
Examples:

- Division by zero
- Null values
- Invalid data type
- Boundary overflow
- Empty input

It is informal but effective when performed by experienced testers.

TOOL USED: KLAROS – TEST MANAGEMENT
Klaros is a web-based application used for managing test cases and test execution.
Activities Performed:

1. Created a new project
2. Defined systems under test
3. Created test environments
4. Designed test cases
5. Created test suites
6. Executed test cases
7. Viewed reports and dashboard

The tool maintains:

- Test case records
- Execution history
- Pass/Fail statistics
- Project progress reports

PROCEDURE:

1. Study the module specification.
2. Identify input domain.
3. Divide input domain using Equivalence Class Partitioning.
4. Apply Boundary Value Analysis.
5. Design test cases.
6. Create a project in Klaros.
7. Add test cases and test suites.
8. Execute test cases manually.
9. Record results.
10. Generate reports.

RESULT: Black box functional test cases were successfully designed using Equivalence Class Partitioning, Boundary Value Analysis, Cause and Effect Graphing, and State Transition Testing techniques. Test execution was performed using Klaros tool and results were recorded.

CONCLUSION: Through this experiment, I gained a clear understanding of black box testing techniques and their importance in functional validation. Equivalence Class Partitioning helped reduce the number of test cases while ensuring adequate coverage. Boundary Value Analysis improved defect detection at input limits. Cause and Effect Graphing was useful for logical condition handling, and State Transition Testing helped validate state-based behavior. The Klaros tool assisted in organizing and managing test cases efficiently. Overall, the experiment enhanced my practical knowledge of functional testing.