Lab 2: Basics of Black Box Testing, Types of Black Box Testing and Boundary Value Analysis

Objective:

- Understand the concept of black box testing.
- Explore various types of black box testing techniques.
- Apply boundary value analysis (BVA) for effective test case design.

Theory:

1. Black Box Testing:

Black box testing is a software testing technique that focuses on testing the functionality of an application without peering into its internal structures or workings. The tester is only concerned with the inputs and outputs of the software system.

Key Features:

- Tester does not require knowledge of internal code or architecture.
- Focuses on functional requirements.
- Validates behavior based on specifications.

2. Types of Black Box Testing:

Type	Description
Functional Testing	Ensures the software performs its intended functions.
Non-functional Testing	Checks performance, usability, reliability, etc.
Regression Testing	Ensures new changes haven't affected existing features.
Smoke Testing	Basic tests to verify critical functionality before further testing.
Sanity Testing	Focuses on one or few areas of functionality after minor changes.
User Acceptance Testing	Conducted to determine if the system meets business needs.
Compatibility Testing	Tests software on different devices, OS, or networks.

3. Boundary Value Analysis (BVA):

Boundary Value Analysis is a test case design technique that focuses on values at the boundaries of input domains rather than in the center.

Example:

For a field that accepts values from 1 to 100:

- Valid boundaries: 1, 100

- Invalid boundaries: 0, 101
- Typical test cases: 0, 1, 2, 99, 100, 101

Lab Requirements:

- Programming Language: Python / Java / C++
- IDE: Any (IDLE, Eclipse, Code::Blocks, etc.)
- Test Case Template (provided below)

Experiment 1: Black Box Testing Example

Problem Statement:

Write a program that determines whether a number is even or odd. Then perform black box testing on it.

Sample Code (Python):

```
def even_or_odd(number):
    if number % 2 == 0:
        return "Even"
    else:
        return "Odd"
```

Test Cases:

Test Case No.	Input	Expected Output	Actual Output	Result
TC1	2	Even	Even	Pass
TC2	5	Odd	Odd	Pass
TC3	0	Even	Even	Pass
TC4	-3	Odd	Odd	Pass

Experiment 2: Boundary Value Analysis

Problem Statement:

Assume a system accepts an input value between 10 and 100. Use BVA to design and test cases.

Sample Code (Python):

```
def is_valid_input(n):
    if 10 <= n <= 100:
        return "Valid"
    else:
        return "Invalid"
```

BVA Test Cases:

Test Case No.	Input	Expected Output	Actual Output	Result
TC1	9	Invalid	Invalid	Pass
TC2	10	Valid	Valid	Pass
TC3	11	Valid	Valid	Pass

TC4	99	Valid	Valid	Pass
TC5	100	Valid	Valid	Pass
TC6	101	Invalid	Invalid	Pass

Result:

- Students understood and implemented black box testing.
- Test cases were designed for both functional and boundary value conditions.

Conclusion:

Black box testing helps testers validate software without internal knowledge of code. Boundary value analysis effectively identifies errors at the limits of input domains.