SET-221
Software Testing Technologies

LAB # 02

LAB Title

Writing and Executing Test cases

Assessment of CLO: 04, PLO: 03

Student Name:			
Roll No.			
Semester		Session	

S. No.	Marks Criteria / Rubrics	Total Marks	Marks Obtained
1	Experiment/Project Execution – Successfully implements the Lab (programming, simulation, circuit design or hardware setup) with correct methodology.	2	
2	Accuracy of Results & Debugging – Obtains correct outputs (program/Code results, waveforms, measurements) with minimal errors and demonstrates debugging skills.	2	
3	Analysis & Interpretation – Compares results with theoretical expectations, identifies errors, and explains deviations logically.	2	
4	Observation & Adaptability – Responds effectively to unexpected challenges, adjusts configurations, and troubleshoots issues in real-time.	2	
5	Report Quality & Documentation – Clearly presents findings, including structured reports with diagrams, code snippets (for software), schematics (for Circuits), or explanations.	2	
	Marks Obtained	10	

Experiment evaluated by

Instructor's Name	Engr.Bushra Aziz		
Date		Signature	

Lab Experiment 2: Writing and Executing Test cases

Objective: To learn how to design and write effective test cases for software applications.

Theory:

Software testing is a crucial part of the software development lifecycle. Well-written test cases are essential for ensuring the quality, reliability, and functionality of software. This lab will guide you through the process of creating comprehensive test cases.

What is a Test Case?

A test case is a specific scenario designed to verify that a particular feature or functionality of a software application works as intended. It includes detailed steps, input data, expected results, and other relevant information.

Why are Test Cases Important?

- **Identify Defects:** Test cases help uncover bugs and defects early in the development process, reducing the cost and effort of fixing them later.
- **Ensure Quality:** They provide a structured way to verify that the software meets the specified requirements.
- **Improve Reliability:** Comprehensive testing increases confidence in the software's reliability and stability.
- **Facilitate Regression Testing:** Test cases can be reused to ensure that new changes or updates haven't introduced new issues.
- **Provide Documentation:** Well-documented test cases serve as a valuable resource for understanding the software's functionality.

Components of a Test Case:

A well-structured test case typically includes the following components:

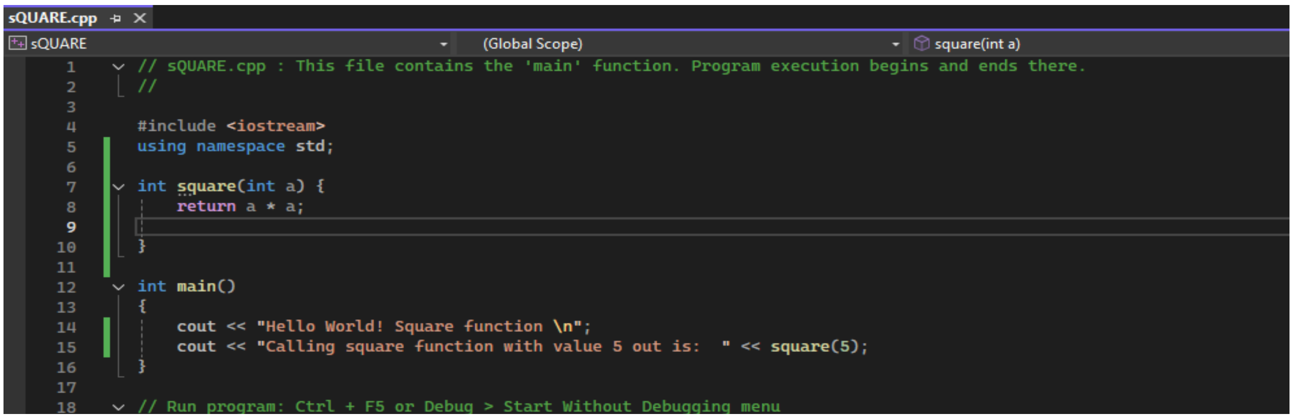
1. **Test Case ID:** A unique identifier for the test case (e.g., TC_001).
2. **Test Case Name:** A descriptive name that clearly indicates the purpose of the test case.
3. **Test Objective/Purpose:** A brief description of what the test case aims to achieve.
4. **Preconditions:** The conditions that must be met before the test case can be executed (e.g., specific data setup, user login).
5. **Test Steps:** A detailed sequence of actions to be performed.
6. **Input Data:** The data used as input for the test case.
7. **Expected Result:** The outcome that should occur if the software functions correctly.
8. **Actual Result:** The actual outcome observed during test execution.
9. **Status:** The result of the test case execution (e.g., Pass, Fail, Blocked).
10. **Tester Name:** The name of the person who executed the test.
11. **Date:** The date of test execution.
12. **Remarks/Comments (Optional):** Any additional notes or observations related to the test case.

Example: Testing a square Function

Let's add an example of testing a square function.

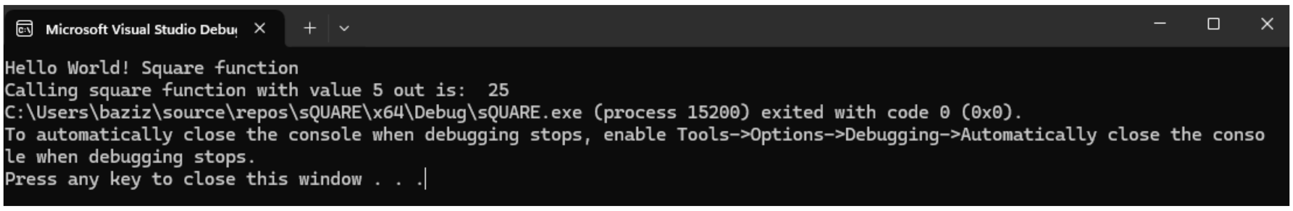
Let's say we have a simple function called **square (a)** that calculates the square of a given number x. We want to write test cases to ensure this function works correctly.

1. Calling square with a value : a=5



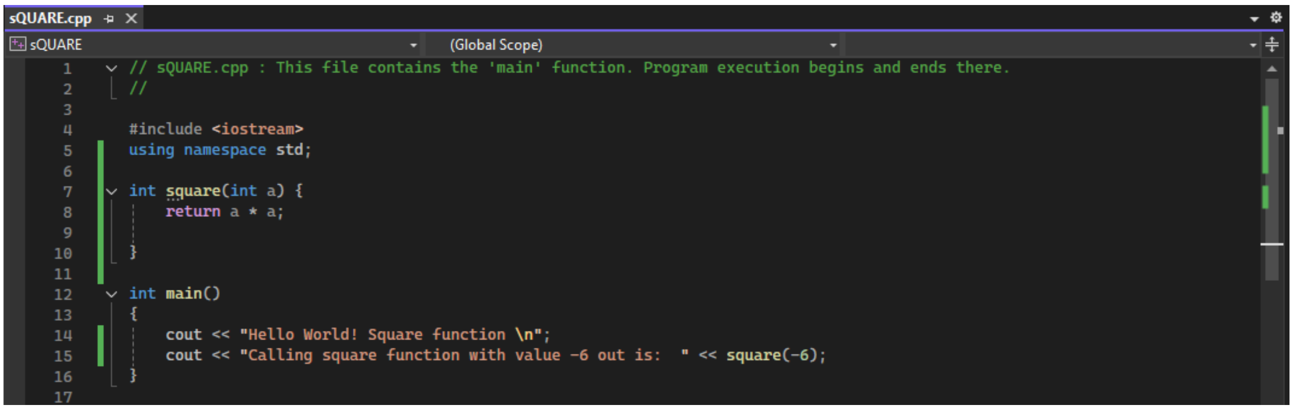
```cpp
// sQUARE.cpp : This file contains the 'main' function. Program execution begins and ends there.
//

#include <iostream>
using namespace std;

int square(int a) {
    return a * a;

}

int main()
{
    cout << "Hello World! Square function \n";
    cout << "Calling square function with value 5 out is:  " << square(5);
}

// Run program: Ctrl + F5 or Debug > Start Without Debugging menu
```

Output



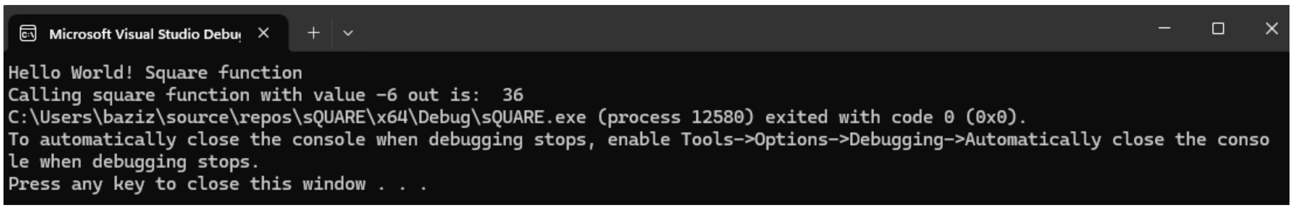
```
Hello World! Square function
Calling square function with value 5 out is:  25
C:\Users\baziz\source\repos\sQUARE\x64\Debug\sQUARE.exe (process 15200) exited with code 0 (0x0).
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the conso
le when debugging stops.
Press any key to close this window . . .
```

2. Calling square with a value : a= -6



```cpp
// sQUARE.cpp : This file contains the 'main' function. Program execution begins and ends there.
//

#include <iostream>
using namespace std;

int square(int a) {
    return a * a;

}

int main()
{
    cout << "Hello World! Square function \n";
    cout << "Calling square function with value -6 out is:  " << square(-6);
}
```

Output

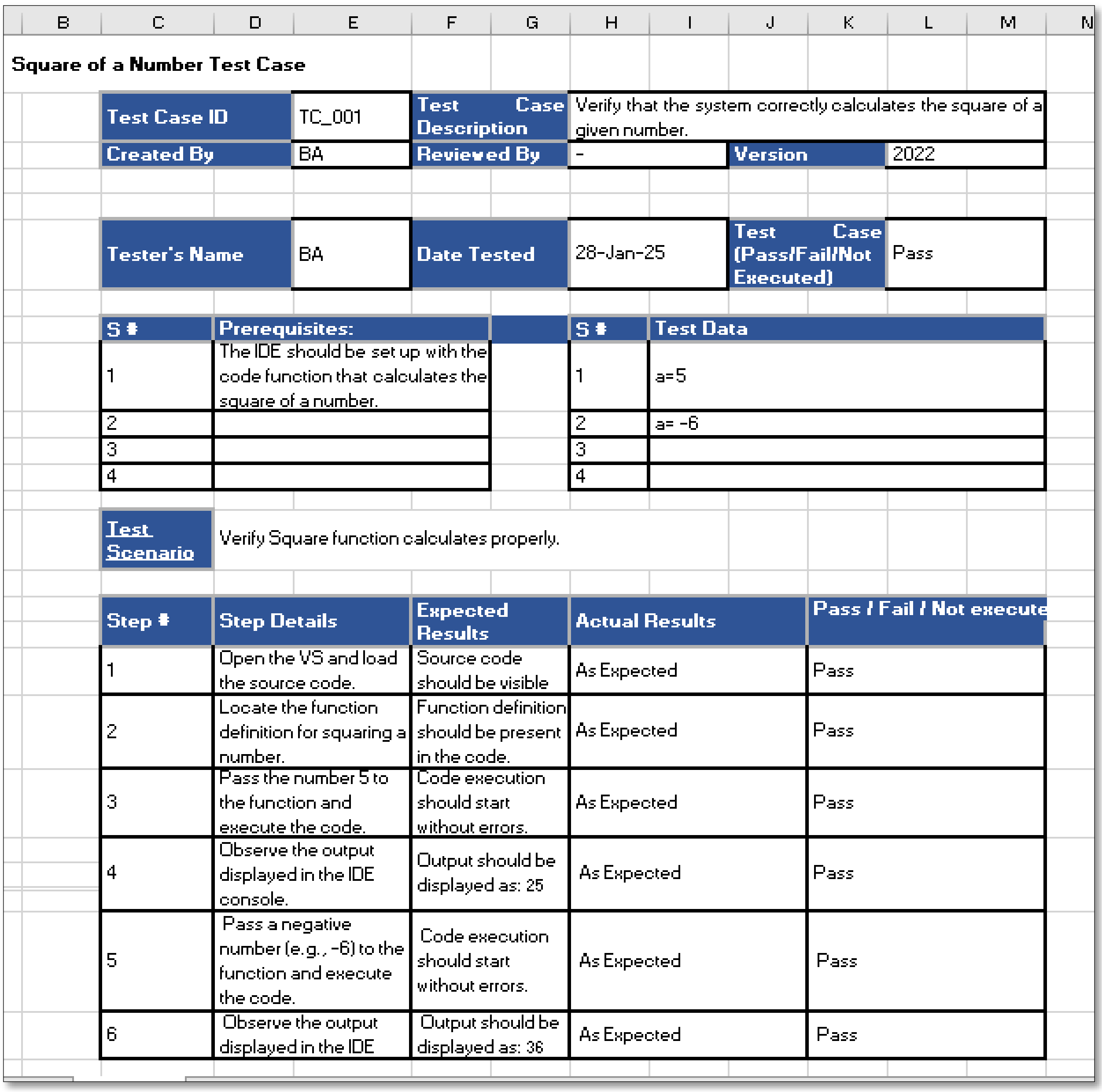


```
Hello World! Square function
Calling square function with value -6 out is:  36
C:\Users\baziz\source\repos\sQUARE\x64\Debug\sQUARE.exe (process 12580) exited with code 0 (0x0).
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the conso
le when debugging stops.
Press any key to close this window . . .
```

Writing Test Cases:

Using test case samples can be beneficial in ensuring consistency and completeness in test case documentation. A well-structured sample provides a clear format to follow, reducing ambiguity and errors in test design. By referencing existing test cases, testers can maintain uniformity, identify missing steps, and streamline the testing process. Additionally, samples serve as a knowledge base for new testers, helping them quickly understand and write effective test cases.

Square of a Number Test Case



Square of a Number Test Case

Test Case ID	TC_001	**Test Case Description**	Verify that the system correctly calculates the square of a given number.		
Created By	BA	**Reviewed By**	-	**Version**	2022

Tester's Name	BA	**Date Tested**	28-Jan-25	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	Prerequisites:
1	The IDE should be set up with the code function that calculates the square of a number.
2	
3	
4	

S #	Test Data
1	a=5
2	a= -6
3	
4	

Test Scenario Verify Square function calculates properly.

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not execute
1	Open the VS and load the source code.	Source code should be visible	As Expected	Pass
2	Locate the function definition for squaring a number.	Function definition should be present in the code.	As Expected	Pass
3	Pass the number 5 to the function and execute the code.	Code execution should start without errors.	As Expected	Pass
4	Observe the output displayed in the IDE console.	Output should be displayed as: 25	As Expected	Pass
5	Pass a negative number (e.g., -6) to the function and execute the code.	Code execution should start without errors.	As Expected	Pass
6	Observe the output displayed in the IDE	Output should be displayed as: 36	As Expected	Pass

Lab Tasks:

1. Write manual test cases for testing an addition function program from Lab01 (Task 2). Consider different test scenarios including positive numbers, negative numbers, and zero.
2. Design and write test cases for checking the login functionality of a UIT'S Edu Smart portal. Use your ID and Passwords for login.

Note: Students should refer to the provided test case sample as a guide while writing their own test cases to ensure clarity, completeness, and consistency.