

**SET-221**

**Software Testing Technologies**

**LAB # 04**

**LAB-TITLE**

Writing Assertions in Google Test

Assessment of CLO: 04, PLO: 03

|  |  |  |  |
| --- | --- | --- | --- |
| Student Name: | AbdullahMohsin | | |
| Roll No. | 23fa-048-st | | |
| Semester | 4th | Session | SPRING-2025 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **S.**  **No.** | **Perf. Level**  **Criteria** | **Excellent (2.5)** | **Good (2)** | **Satisfactory (1.5)** | **Needs Improvement (0 ~ 1)** | **Marks Obtained** |
| **1** | Project Execution & Implementation | Fully functional, optimized, and well-structured. | Minor errors, mostly functional. | Some errors, requires guidance. | Major errors, non-functional,  or not Performed. |  |
| **2** | Results & Debugging Or  Troubleshooting | Accurate results with effective debugging  Or Troubleshooting. | Mostly correct, some debugging Or Troubleshooting needed. | Partial results, minimal debugging  Or Troubleshooting. | Incorrect results, no debugging Or Troubleshooting, or not attempted. |  |
| **3** | Problem- Solving & Adaptability (VIVA) | Creative approach, efficiently solves challenges. | Adapts well, minor struggles. | Some adaptability, needs guidance. | Lacks innovation or no innovation, unable to solve problems. |  |
| **4** | Report Quality & Documentation | Clear, structured, with detailed visuals. | Mostly clear, minor gaps. | Some clarity issues, missing details. | Poorly structured, lacks  clarity, or not submitted. |  |
| **Total Marks Obtained Out of 10** | | | | | |  |

Experiment evaluated by

|  |  |  |  |
| --- | --- | --- | --- |
| Instructor’s Name | Engr.Bushra Aziz | | |
| Date |  | Signature |  |

Copyright © Department of Engineering & Technology – UIT University Karachi

**Objective:** Exploring various types of assertions in Google Test and Learn how to apply assertions to test C++ code.

# Lab Exercise

1. Write a test case for modulus and power function.

Ans:

#include "pch.h"

#include "C:\Users\23fa-048-st\Desktop\PROJECT\app\app\cal.cpp"

TEST(CalculatorTest, Addition) {

ASSERT\_EQ(add(2, 3), 5);

ASSERT\_NE(add(4, 4), 9);

ASSERT\_GT(add(3, 2), 4);

}

TEST(CalculatorTest, Subtraction) {

ASSERT\_EQ(subtract(10, 5), 5);

ASSERT\_LE(subtract(3, 3), 0);

ASSERT\_LT(subtract(2, 5), 0);

}

TEST(CalculatorTest, Multiplication) {

ASSERT\_EQ(multiply(3, 4), 12);

ASSERT\_NE(multiply(-2, 5), 10);

ASSERT\_GE(multiply(5, 2), 10);

}

TEST(CalculatorTest, Modulus) {

ASSERT\_EQ(modulus(10, 3), 1);

ASSERT\_EQ(modulus(20, 5), 0);

ASSERT\_NE(modulus(15, 4), 3);

}

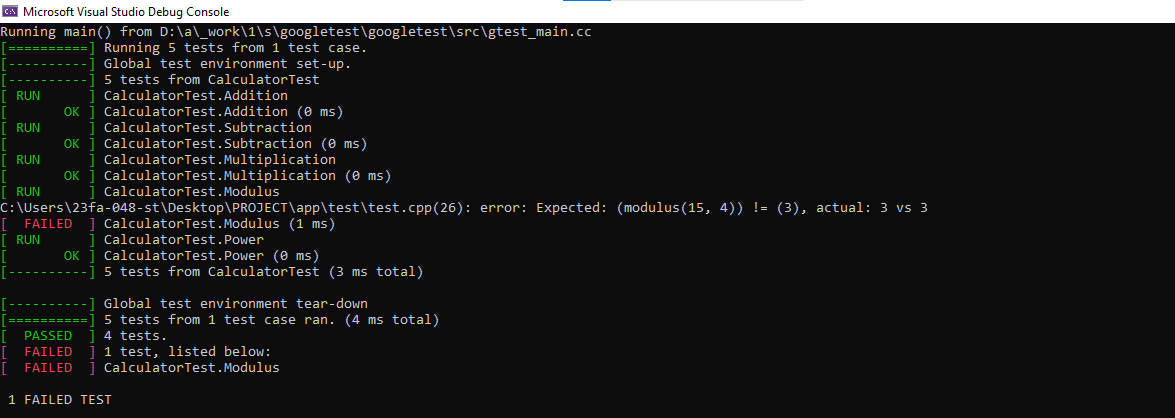
TEST(CalculatorTest, Power) {

ASSERT\_DOUBLE\_EQ(power(2, 3), 8.0);

ASSERT\_DOUBLE\_EQ(power(5, 0), 1.0);

ASSERT\_NEAR(power(9, 0.5), 3.0, 0.0001);

}



1. Add more edge cases for division, including large numbers and floating-point precision.

Ans:

#include "pch.h"

#include "C:\Users\23fa-048-st\Desktop\PROJECT\app\app\cal.cpp"

TEST(CalculatorTest, Addition) {

ASSERT\_EQ(add(2, 3), 5);

ASSERT\_NE(add(4, 4), 9);

ASSERT\_GT(add(3, 2), 4);

}

TEST(CalculatorTest, Subtraction) {

ASSERT\_EQ(subtract(10, 5), 5);

ASSERT\_LE(subtract(3, 3), 0);

ASSERT\_LT(subtract(2, 5), 0);

}

TEST(CalculatorTest, Multiplication) {

ASSERT\_EQ(multiply(3, 4), 12);

ASSERT\_NE(multiply(-2, 5), 10);

ASSERT\_GE(multiply(5, 2), 10);

}

TEST(CalculatorTest, Modulus) {

ASSERT\_EQ(modulus(10, 3), 1);

ASSERT\_EQ(modulus(20, 5), 0);

ASSERT\_NE(modulus(15, 4), 3);

}

TEST(CalculatorTest, Power) {

ASSERT\_DOUBLE\_EQ(power(2, 3), 8.0);

ASSERT\_DOUBLE\_EQ(power(5, 0), 1.0);

ASSERT\_NEAR(power(9, 0.5), 3.0, 0.0001);

}

TEST(CalculatorTest, Division) {

ASSERT\_DOUBLE\_EQ(divide(10, 2), 5.0);

ASSERT\_NEAR(divide(7, 2), 3.5, 0.01);

ASSERT\_TRUE(divide(9, 3) == 3.0);

ASSERT\_DOUBLE\_EQ(divide(1e9, 1e3), 1e6);

ASSERT\_NEAR(divide(1.0, 3.0), 0.3333, 0.0001);

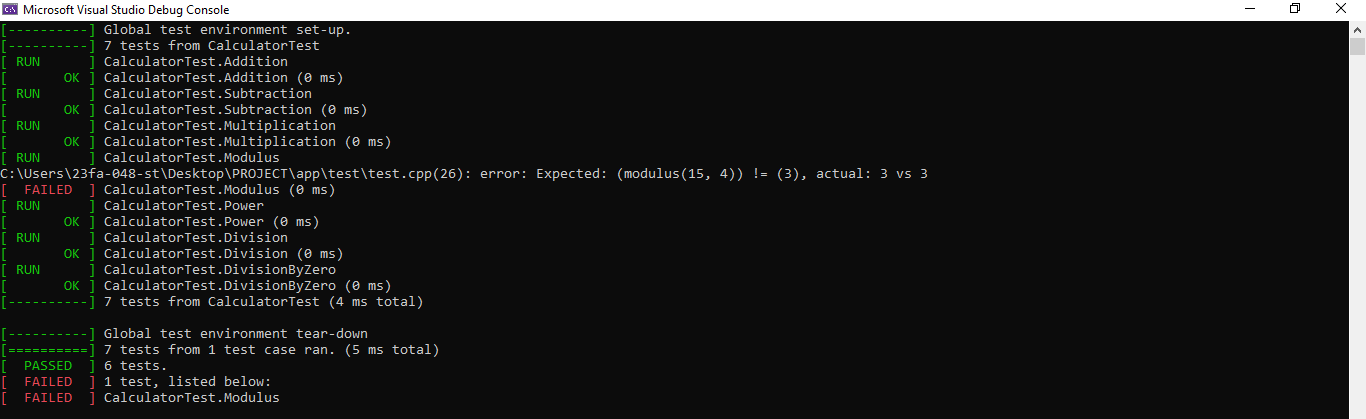
}

TEST(CalculatorTest, DivisionByZero) {

ASSERT\_THROW(divide(5, 0), std::runtime\_error);

ASSERT\_ANY\_THROW(divide(1, 0));

}



1. Write any program that contain string data then test it using different string assertions such as ASSERT\_STREQ and ASSERT\_STRNE.

#include "pch.h"

#include "C:\Users\23fa-048-st\Desktop\PROJECT\app\app\cal.cpp"

TEST(CalculatorTest, Addition) {

ASSERT\_EQ(add(2, 3), 5);

ASSERT\_NE(add(4, 4), 9);

ASSERT\_GT(add(3, 2), 4);

}

TEST(CalculatorTest, Subtraction) {

ASSERT\_EQ(subtract(10, 5), 5);

ASSERT\_LE(subtract(3, 3), 0);

ASSERT\_LT(subtract(2, 5), 0);

}

TEST(CalculatorTest, Multiplication) {

ASSERT\_EQ(multiply(3, 4), 12);

ASSERT\_NE(multiply(-2, 5), 10);

ASSERT\_GE(multiply(5, 2), 10);

}

TEST(CalculatorTest, Modulus) {

ASSERT\_EQ(modulus(10, 3), 1);

ASSERT\_EQ(modulus(20, 5), 0);

ASSERT\_NE(modulus(15, 4), 3);

}

TEST(CalculatorTest, Power) {

ASSERT\_DOUBLE\_EQ(power(2, 3), 8.0);

ASSERT\_DOUBLE\_EQ(power(5, 0), 1.0);

ASSERT\_NEAR(power(9, 0.5), 3.0, 0.0001);

}

TEST(CalculatorTest, Division) {

ASSERT\_DOUBLE\_EQ(divide(10, 2), 5.0);

ASSERT\_NEAR(divide(7, 2), 3.5, 0.01);

ASSERT\_TRUE(divide(9, 3) == 3.0);

ASSERT\_DOUBLE\_EQ(divide(1e9, 1e3), 1e6);

ASSERT\_NEAR(divide(1.0, 3.0), 0.3333, 0.0001);

}

TEST(CalculatorTest, DivisionByZero) {

ASSERT\_THROW(divide(5, 0), std::runtime\_error);

ASSERT\_ANY\_THROW(divide(1, 0));

}

TEST(CalculatorTests, TestName) {

ASSERT\_STREQ("LabFour", name().c\_str());

ASSERT\_STRNE("WrongName", name().c\_str());

}

