# Exercise: NUnit Unit Testing - Calculator Addition

## Objective

To implement and validate unit testing for a calculator’s addition method using NUnit framework.   
This includes using TestFixture, SetUp, TearDown, Test, and TestCase attributes to ensure correctness and modularity of tests.

## Unit Testing vs Functional Testing

- Unit Testing: Tests individual components (e.g., methods) in isolation.  
- Functional Testing: Tests the system's functionality end-to-end based on requirements.  
- Unit tests are faster and easier to debug; functional tests simulate real user scenarios.

## Key NUnit Attributes Used

- [TestFixture]: Denotes a class that contains NUnit tests.  
- [SetUp]: Runs before each test method to initialize objects.  
- [TearDown]: Runs after each test to clean up resources.  
- [TestCase]: Defines multiple sets of inputs and expected outputs for one method.  
- Assert.That: Validates that the actual result matches the expected result.

## Benefits of Parameterized Tests

- Avoids code duplication by testing multiple input/output combinations in a single method.  
- Makes tests cleaner and easier to manage.  
- Ensures wider coverage for edge cases with minimal effort.

## Test Result Screenshot

The following screenshot shows successful execution of 3 NUnit test cases for the addition method.

