# Technical Competency Test

This is a set of progressively more difficult steps that aims to cover the technology knowledge, methodology experience and related skills expected of a senior software engineer in the TMP team.

Consider the following simple C# interface:



## Visual Studio

1. Create an empty solution called CalculatorTest.
2. Create a class library containing the interface above.
3. Create a C# class to realize the interface as a C# class.

## TDD

1. Create a class library containing a suite of unit tests for the interface using test-driven design principles.

## C#

1. Add a diagnostics interface to the calculator class to allow each method to report its calculation results to the *console*.
2. Mock the diagnostics interface and use it to refactor the unit tests so that the test suite checks that the diagnostics interface is called with expected values.
3. Implement a simple console application using Visual Studio that will use the class.

## IoC / Dependency Injection

1. Implement the diagnostics interface and inject it into the calculator at runtime.
2. Implement a dummy diagnostics interface that doesn’t report anything.

## Entity Framework / SQL Server

1. Using a SQL Server Database, create an Entity Framework ORM layer to allow the console application to write the diagnostics information to the database.
2. Implement a class that realizes the diagnostics interface and modify the console application so that it outputs the diagnostics to the database using EF.
3. Implement another class that does not use EF and instead uses a stored procedure and ADO.NET to write the diagnostics data.

## Web Services

1. Create a simple web service that provides access to the calculator class via a REST API using HTTP.
2. Modify the console application to access the calculator web service to perform its calculations instead of using an internal class.