# Student Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Due Date: **Sunday, Oct. 10th @ 11:59pm**

# Lab 2

The goal of this assignment is to familiarize you with Unit Testing in C#. This assignment relies on you following the **C# Test Driven Development** course on LinkedIn Learning (<https://www.linkedin.com/learning/c-sharp-test-driven-development-2/welcome?u=2149225>). You have an account with LinkedIn Learning associated with your NSCC account: See <https://subjectguides.nscc.ca/LinkedinLearning/getstarted>.

**NOTE:** This lab involves going through the first 4 modules of the online course on your own, while completing the extra exercises that I have added below. I recommend watching a few videos, then do the exercises I provide before continuing on to the next section/modules. Some of the modules in the course have Challenges that we will build on directly below. Otherwise, we will just build on the examples provided in the video. You will demonstrate your solutions to these exercises to me as part of a code review. Be prepared to answer questions about the code and how you accomplished the exercise. Each item will have a score associated with it which will accumulate to your final mark for this lab.

**Note:** Please organize your code so that each numbered section of the assignment is its own folder and Visual Studio project within your github repository. For example:

* Lab2
  + Module1
  + Module2

I recommend copying and renaming previous projects to build on what you did already. You may copy the resource files from LinkedIn Learning as a starting point. However, I recommend copying the starting files and not the solution so you get a chance to work through the code as it is explained in the video.

## Module 1 (A Simple Test Driven C# Project)

Finish Sections 1-3 in the LinkedIn Learning Course. Then copy in the new unit tests provided with this lab document: Labs\Lab2\Module1\SalaryCalculatorTestProject\Calculator\CalculatorTests.cs in Lab2\_Code\_CalculatorTests.zip.

1. Get both Unit tests passing for negative and zero values in the GetHourlyWage() and GetAnnualSalary methods. Note this is a "reversed test" in that you expect an exception to be thrown and your Assert is on the error message received.

**(2 pts – One for each method implemented) \_\_\_\_\_**

1. Uncomment and add the remaining unit tests and change SalaryCalculator.cs (your Calculator class) to meet the following requirements:

**Add a function that calculates the total amount of tax withheld from an employee’s weekly salary.**

**The total withheld tax amount is calculated by combining the amount of provincial tax withheld and the amount of federal tax withheld, minus a per-dependent deduction from the total tax withheld. The user will enter their pre-tax weekly salary amount and the number of dependents they wish to claim. The program will calculate and output the amount of provincial tax withheld, amount of federal tax withheld, the dependent tax deduction, and the user’s final take-home amount.**

**Provincial withholding tax is calculated at 6.0%. Federal withholding tax is calculated at 25.0%. The tax deduction for dependents is calculated at 2.0% of the employee’s salary per dependent**.

Your **TaxWitheld** function will take two parameters: **weeklySalary** [double] and **numDependents** [int]. The function will return a POCO called **TaxData** with five double parameters, one for each of the required values. For example, calling **TaxWithheld(1000, 2)** should return a TaxData with the following properties and values:

TaxData {

ProvincialTaxWithheld: 60.0

FederalTaxWithheld: 250.0

DependentDeduction: 40.0

TotalWithheld: 270.0

TotalTakeHome: 730.0

}

**Note:** **TaxData** should be in its own .cs file in the Calculator project.

**(1 pt – TaxData Class in own file in Calculator Project NOT Test Project) \_\_\_\_\_**

**(4 pts – 1 point for each provided weeklySalary Unit Test passing) \_\_\_\_\_**

## Module 2 (An existing C# Project)

Finish chapter 4 (An Existing C# Project) of the LinkedIn Learning Course. Copy the folder of the final solution into Labs\Lab2\Module2\ in your repo. Update the Polymorphism project to do the following:

* Add a new public Interface called **IWorker** with a **CalculateWeeklySalary** method with no body. The interface must be in its own file called IWorker.cs. (**1pt)** \_\_\_\_\_
* Decouple Employee and Contractor to no longer be in a base-subclass relationship. Instead, make them both implement **IWorker**, each **CalculateWeeklySalary** method no longer needs override or virtual. (**1pt)** \_\_\_\_\_
* Update the mock code to handle a list of IWorkers and not Employees. Demonstrate that all unit tests pass, despite changing the underlying implementation. (**1pt)** \_\_\_\_\_

# Final Score (Out of 10) **\_\_\_\_\_\_\_**