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

# Assignment 1

The goal of this assignment is to familiarize you with Object Oriented Programming in C#, git and Visual Studio. Later assignments in the course will be less structured than this one as this assignment relies on you following the **Learning C#** course on LinkedIn Learning (<https://www.linkedin.com/learning/learning-c-sharp-3/welcome?u=2149225>). You have an account with LinkedIn Learning associated with your NSCC account: See <https://subjectguides.nscc.ca/LinkedinLearning/getstarted>.

Essentially, this Assignment involves going through the first 4 modules of the **Learning C#** 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 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 these exercises to me in class 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.

**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:

* Assignment1
  + Module1
    - Section1
    - Section2

I recommend copying and renaming previous projects to build on what you did already. to You may copy the resource files from Lynda.com 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.

## General

1. Demonstrate that you have made multiple commits as you went through the **Learning C#** course: at least 5. **(3 pts) \_\_\_\_\_**
2. Show that you have the right README and .gitignore for Visual Studio. Also, arrange your project according to the structure outlined above. **(2 pts) \_\_\_\_\_**

## Module 1 (Basics)

1. Take the Challenge 1 (Multiple Choice) and add two more questions of your choosing to it.

**(1 pt) \_\_\_\_\_**

1. Change Challenge 2 (Passcode) so that the user can change the password after authenticating. **Question:** Is an else...if the best option for checking the secret? Simplify the code to not use else…if as described. **(2 pts) \_\_\_\_\_**
2. Change Challenge 3 (Loop Counting) to be only 2 loops with a conditional. **(2 pts) \_\_\_\_\_**
3. How would you change Challenge 4 (Student Manager) to be a single two dimensional array? Each row in the main array is itself an array of two elements: one for name, one for grade. **Hint:** A 2D array is declared in C# as follows **var array = new string[<size>,<size>]**. You may remove the need for parsing ints for this exercise and store everything as a string. **(2 pts) \_\_\_\_\_**

## Module 2 (OOP in C#)

1. Challenge 5:
   * Make sure you use private member variables and get/set pairs for the Data object.
   * Change the Survey console app to be a full Zodiac Calculator. Requirements: Also take the day of the month as an input. Use a switch statement to calculate what star sign matches the month and date. You should validate that the user entered a number for month and date. **(4 pts) \_\_\_\_\_**

## Module 3 Exception Handling & Debugging

1. Enhance the StudentManager program to have a custom error class that extends FormatException. Make sure the main program code correctly handles the custom error class and reports the proper error message. **(2 pts) \_\_\_\_\_**
2. Demonstrate your knowledge of the debugger to your instructor. **(2 pts) \_\_\_\_\_**
   1. Comment out any try catch and force a debug on exception.
   2. Re-instate the commented out code and demonstrate the following:
      1. A breakpoint
      2. Step Over
      3. Step Into

## Module 4

1. **(Enums – Module 4-1)** - Enhance the survey code to use an enum for the months: Jan = 0, Feb = 1, etc. Show it in the switch as well. **(2 pts) \_\_\_\_\_**
2. **(out vs ref – Module 4 (sects 2-5))** – Update the outreference class to have a static parseDouble function. This will take an input String as an out parameter and return it as a double to the calling function. The function will return true if parsed successfully or false if not. In the Main method provide 2 calls one that sets the value properly, another which passes in your name and returns false. See below for the calling code: **(2 pts) \_\_\_\_\_**

var test = "1.5";

var outDouble = 0.0;

Console.WriteLine("Result: {0}", parseDouble(test, out outDouble));

Console.WriteLine("Test type is {0}", outDouble.GetType());

test = "Ronan";

outDouble = 0.0;

Console.WriteLine("Result: {0}", parseDouble(test, out outDouble));

Console.WriteLine("Test value is {0}", outDouble);

1. **(Interfaces – Module 4(sect 7))** – After completing the code in this section, extend by doing the following: move Student & Member to their own class files. Next, create an interface called IGraduate with a method of graduate(). Make student implement IGraduate as well as extend Member. Make sure the graduate() method prints to the console when called. Add a new class called Registrar that takes the list of students as <IGraduate>s and calls the graduate() method on each one in a graduateAll() method. This should be called at the end of the Program main method. The constructor for the Registrar class should look like the following: **(4 pts) \_\_\_\_\_**

public Registrar(List<IGraduate> grads)

{

this.grads = grads;

}

1. **(Events – Module 4(sect 8))** – After completing the code in this section, extend by doing the following: The marketing department also wants to be notified when the Survey is completed. Add a second class called Marketing that will act as a listener to the Posted event that will print to the console - "Thank you for completing the survey. You are now subscribed to ten of our newsletters." Demonstrate to your instructor. **(2 pts) \_\_\_\_\_**
2. **(Abstract Classes – Module 4(sect 9))** After completing the code in this section, extend by doing the following: Add a Circle shape to the GeometryTool application. Demonstrate by instantiating a Circle with a radius of 3. It should inherit from the abstract Shape class and override the GetArea function and display its results using circle.Display(). **Note:** Make sure you use the Math.PI constant. You should also upgrade the Shape class to handle doubles and not ints.

**(4 pts) \_\_\_\_\_**

# Final Score (Out of 34) **\_\_\_\_\_\_\_**