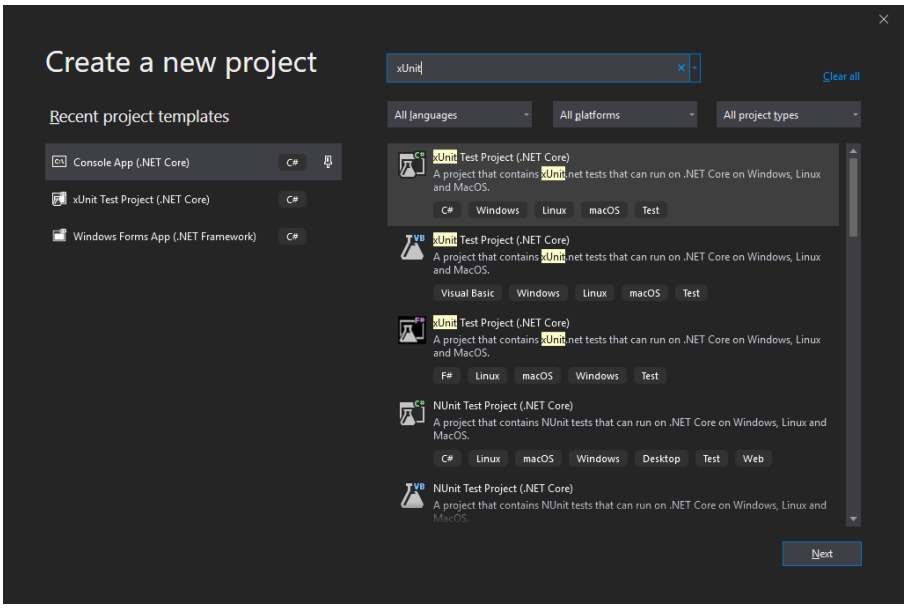
Set Up Unit Testing in a Console App

Create a new Console App named FirstUnitTestLastName.

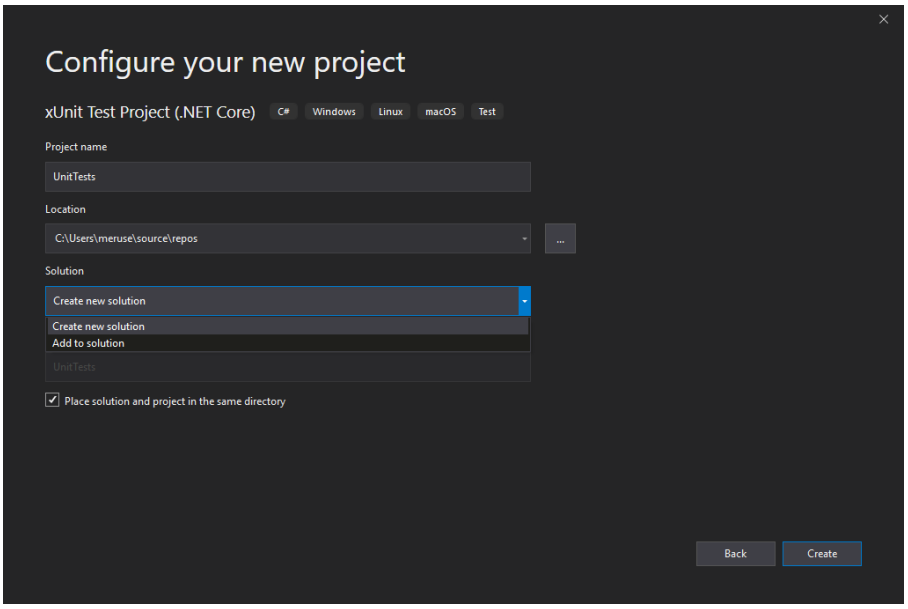
Rename Program.cs to MedalWinnerProgram.cs and inside, rename the namespace to MedalWinner, add pubic access modifier in front of class MedalWinnerProgram

```
namespace MedalWinner
{
    public class MedalWinnerProgram
```

Create a new project in the solution of Type xUnit Test



Name it UnitTests and Add to solution.



Code

Add the following code to MedalWinnerProgram.cs Main method:

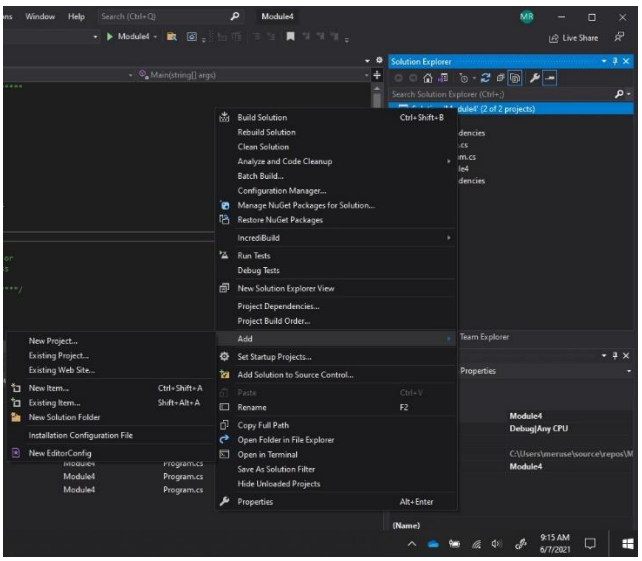
```
static void Main(string[] args)
{
    // Call a method that returns the user's first name
    // first letter capitalized only
    // and the last name all capitalized followed by rank
    string winner1 = MedalWinner("Megan", "rapinoE", "1");
    string winner2 = MedalWinner("USA", "Women's Soccer","1");
    string winner3 = MedalWinner("USA", "Women's Soccer", "1.5");
    Console.WriteLine("And the winner is ... {0:G}", winner1);
    Console.WriteLine("Expected: Megan RAPINOE 1");
    Console.WriteLine("And the winner is ... {0:G}", winner2);
    Console.WriteLine("Expected: Usa WOMEN'S SOCCER 1");
    Console.WriteLine("And the winner is ... {0:G}", winner3);
    Console.WriteLine("Expected: Usa WOMEN'S SOCCER 1.5");

}
```

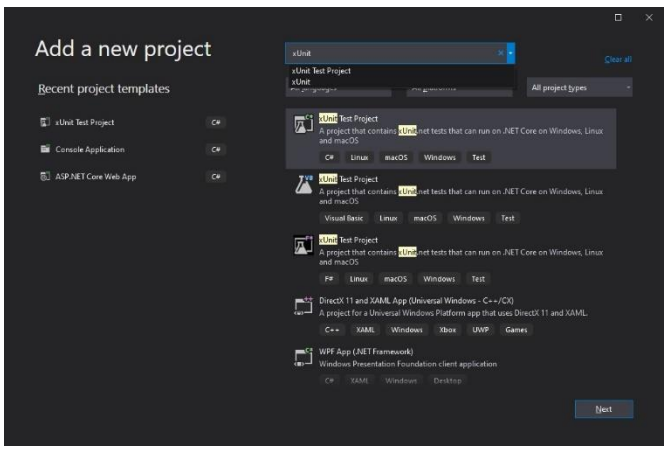
Add the method above Main:

```
public static string MedalWinner(string fName, string lName, int rank)
{
    string result;
    result = fName + " " +  lName + " " + rank.ToString();
    return result;
}
```

Create xUnit Test Project

Right click on the Solution (not project) name, select Add, select New Project …





Search for xUnit Test Project

Tests

Add the following tests to the unit testing class

```
[Fact]
        public void PassingCaseTest()
        {
            // Arrange
            string fName = "Megan";
            string lName = "RAPINOE";
            string rank = "1";

            string expected = "Megan RAPINOE 1";
            string actual;

            // Act
            actual = MedalWinnerProgram.MedalWinner(fName, lName, rank);

            // Assert
            Assert.Equal(expected, actual);
        }
        [Fact]
        public void LastNameFormatTest()
        {
            // Arrange
            string fName = "Megan";
            string lName = "rapinoe";
            string rank = "1";

            string expected = "Megan RAPINOE 1";
            string actual;

            // Act
            actual = MedalWinnerProgram.MedalWinner(fName, lName, rank);

            // Assert
            Assert.Equal(expected, actual);
        }
        [Fact]
        public void FirstNameFormatTest()
        {
            // Arrange
            string fName = "megan";
            string lName = "RAPINOE";
            string rank = "1";

            string expected = "Megan RAPINOE 1";
            string actual;

            // Act
            actual = MedalWinnerProgram.MedalWinner(fName, lName, rank);

            // Assert
            Assert.Equal(expected, actual);
        }
        [Fact]
        public void RankExceptionHandledTest()
        {
```

```
            // Arrange
            string fName = "USA";
            string lName = "Women's Soccer";
            string rank = "1.5";

            string expected = "Usa WOMEN'S SOCCER 1.5";
            string actual;

            // Act
            actual = MedalWinnerProgram.MedalWinner(fName, lName, rank);

            // Assert
            Assert.Equal(expected, actual);

        }
```

Debug

Coding involves a lot of debugging, finding errors and fixing them.

- Fix the unhandled exception by adding code to the method MedalWinner()
- Fix the business logic (first name format and last name format)
- Do not change the method call in the Main
- Take a screen shot of the code with ALL errors fixed

Notice you can use Unit Tests to find and help fix exceptions and logic errors. This is an important part of programming.