# Fibonacci demo

These instructions are for using

* dotnet core
* nUnit

## Person A

* create .NET Core NUnit Test Project - Fibonacci.Tests
* create test fixture - FibonacciEnumeratorShould
* add 1st test - ReturnFirstElementAsOne

*var seq = new FibonacciEnumerator();*

*Assert.AreEqual(1, seq.First());*

* create class for *FibonacciEnumerator : IEnumerable<int>*

## Person B

* pass test

*yield return 1;*

* add 2nd test - ReturnSecondElementAsOne

*var seq = new FibonacciEnumerator();*

*Assert.AreEqual(1, seq.ElementAt(1));*

## Person A

* pass test

*yield return 1;*

* add 3rd test - ReturnThirdElementAsTwo

*Assert.AreEqual(2, seq.ElementAt(2));*

## Person B

* pass test

*yield return 2;*

* refactor tests - repeat of 3 - add params, rename ReturnCorrectElementWhenPositionIsPassed
* add 4th test case

*[TestCase(3, 3)]*

## Person A

* pass test

*yield return 3;*

* refactor with a loop - pattern emerging

*for (var i = 1;; i++)*

*{*

*yield return i;*

*}*

* add 5th test case

*[TestCase(5, 4)]*

## Person B

* pass test
* refactor with proper logic

*var previous = 0;*

*var current = 1;*

*for (var i = 1;; i++)*

*{*

*var next = previous + current;*

*previous = current;*

*current = next;*

*yield return current;*

*}*

* clean up code - change for loop to a while

*while (true)*

* add passing test cases to sequence 10

*[TestCase(8, 5)]*

*[TestCase(13, 6)]*

*[TestCase(21, 7)]*

*[TestCase(34, 8)]*

*[TestCase(55, 9)]*

* add 50th test case

*[TestCase(12586269025, 49*

## Person A

* pass test by updating int to long

*long expected*

*: IEnumerator<long>*

*long previous = 0;*

*long current = 1;*