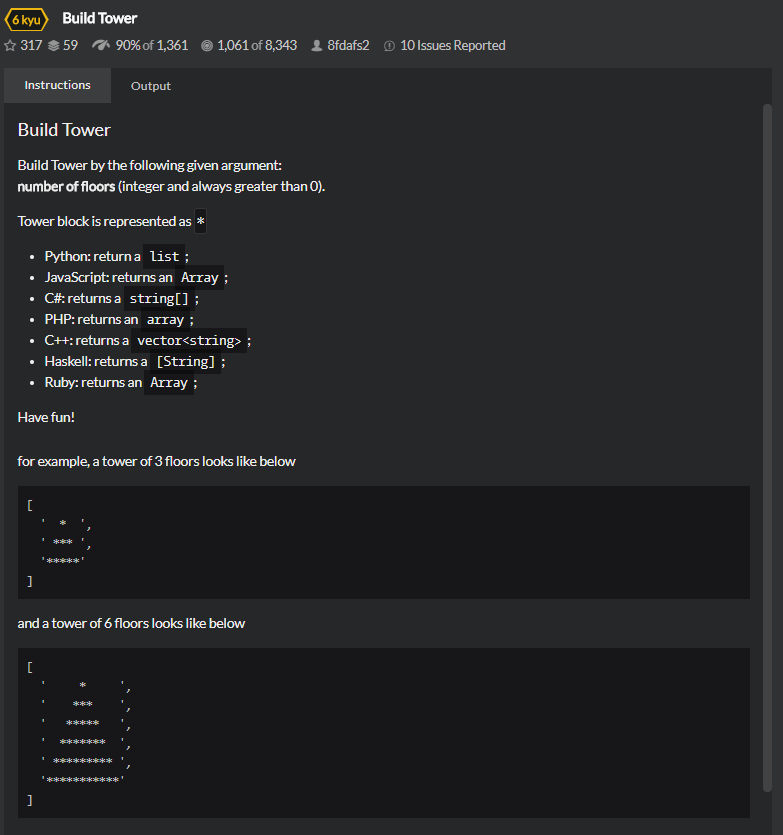
Day28. 把塔蓋起來蓋起來!! Codewars\_Build Tower

今天的題目走懷舊風，懷念一下當時大學寫過的題目(?

好懷念啊(是過多久?

明明大學才剛畢業不到一年啊O\_O

不多說惹，這是今天的題目!



題目需求可以拆分成兩個

1. 輸出星號
2. 輸出空白
3. 合併星號與空白(說白一點就是把兩個Function加起來而已存在一個字串陣列而已)

來吧，召喚一地個Testcase，先來寫輸出星號的Function

```

[TestMethod]

public void PrintStar\_Input\_1\_Should\_Be\_1Star()

{

Assert.AreEqual("\*", Kata.PrintStar(1));

}

```

Production Code Alt+Enter自動產出，如下(老樣子)

```

public static string PrintStar(int n)

{

throw new System.**NotImplementedException**();

}

```

紅燈也是老樣子，Commit，接下來改Production Code

public static string PrintStar(int n)

{

return "\*";

}

Pass&Commit!

再來寫輸入2的測試案例

```

[TestMethod]

public void PrintStar\_Input\_2\_Should\_Be\_3Star()

{

Assert.AreEqual("\*\*\*", Kata.PrintStar(2));

}

```

要輸出奇數的\*就需要想一下奇數的公式是2n-1。

改一下Production Code，就會變成這個樣子。

```

public static string PrintStar(int n)

{

return string.Join("", Enumerable.Repeat("\*", 2 \* n - 1));

}

```

接下來要處理輸出空白的方法囉

一樣，輸入1的測試案例。

```

[TestMethod]

public void PrintSpace\_Input\_1\_Should\_Be\_StringEmpty()

{

Assert.AreEqual(string.Empty,Kata.PrintSpace(1));

}

```

Fail&Commit then Do&Pass then Commit, 以下是Pass輸入1的Production Code。

```

public static string PrintSpace(int n)

{

return string.Empty;

}

```

接下來要考慮到輸入2的test case囉!

```

[TestMethod]

public void PrintSpace\_Input\_2\_Should\_Be\_1Space()

{

Assert.AreEqual(" ", Kata.PrintSpace(2));

}

```

而因為輸入1要返回空的字串，所以我們repeat的range方法需要是n-1

```

public static string PrintSpace(int n)

{

return string.Join("", Enumerable.Repeat(" ", n - 1));

}

```

現在兩個需求都完成了，來寫一個結合兩個需求的Test Code吧!

老樣子，先輸入1

```

[TestMethod]

public void Input\_1\_Should\_Be\_1star()

{

CollectionAssert.AreEqual(new string[] { "\*" }, Kata.TowerBuilder(1));

}

```

接下來就來處理這個測試案例吧!

因為不需要考慮多個陣列元素所以Production Code長這樣。

```

public static string[] TowerBuilder(int n)

{

return new string[] { PrintSpace(n) + PrintStar(n) + PrintSpace(n) };

}

```

再來要考慮到多個陣列元素囉!

所以現在來輸入2

```

[TestMethod]

public void Input\_2\_Should\_Be\_1starAnd3Star()

{

CollectionAssert.AreEqual(new string[]{"\*"," \*\*\* "},Kata.TowerBuilder(2));

}

```

因為要考慮多個陣列元素的輸出，所以這時候要用迴圈來完成輸入2的需求。

Production Code如下

```

public static string[] TowerBuilder(int n)

{

var result = new List<string>();

for (int i = 1; i <= n; i++)

{

result.Add(PrintSpace(i) + PrintStar(i) + PrintSpace(i));

}

return result.ToArray();

}

```

跑個測試，PASS!

現在要考慮到輸入3了，你會發現我這個Code的輸出空白是不符合需求的

所以產生了輸入3的測試案例。

```

[TestMethod]

public void Input\_3\_Should\_Be\_1starAnd3StarAnd5Star()

{

CollectionAssert.AreEqual(new string[] { " \* ", " \*\*\* ", "\*\*\*\*\*" }, Kata.TowerBuilder(3));

}

```

接下來改一下Production Code，這時候我發現怎麼改我的輸入2的測試就是不過，但輸入3的居然過了。

於是重新審視了輸入2的測試案例，發現期望的輸出寫錯了，所以改好之後就All Pass了

Production Code和更改之後的輸入2測試案例

```

[TestMethod]

public void Input\_2\_Should\_Be\_1starAnd3Star()

{

CollectionAssert.AreEqual(new string[] { " \* ", "\*\*\*" }, Kata.TowerBuilder(2));

}

```

```

public static string[] TowerBuilder(int n)

{

var result = new List<string>();

for (int i = 1; i <= n; i++)

{

result.Add(PrintSpace(n - i + 1) + PrintStar(i) + PrintSpace(n - i + 1));

}

return result.ToArray();

}

```

其實會發現PrintSpace的-1跟傳入PrintSpace的+1是很多餘的，所以我們需要重新設計一下PrintSpace的測試案例補上輸入0的測試案例並修改Production Code

，修改完成之後會發現Production Code是可以用Linq完成的。

所以Refactor 之後Pass Unit Test的所有Production Code變成這個樣子

```

public static string PrintStar(int n)

{

return string.Join("", Enumerable.Repeat("\*", 2 \* n - 1));

}

public static string PrintSpace(int n)

{

return string.Join("", Enumerable.Repeat(" ", n));

}

public static string[] TowerBuilder(int n)

{

return Enumerable.Range(1, n).Select(x => PrintSpace(n - x) + PrintStar(x) + PrintSpace(n - x)).ToArray();

}

```

這是今天所有的測試案例

```

[TestClass]

public class UnitTest1

{

[TestMethod]

public void PrintStar\_Input\_1\_Should\_Be\_1Star()

{

Assert.AreEqual("\*", Kata.PrintStar(1));

}

[TestMethod]

public void PrintStar\_Input\_2\_Should\_Be\_3Star()

{

Assert.AreEqual("\*\*\*", Kata.PrintStar(2));

}

[TestMethod]

public void PrintSpace\_Input\_0\_Should\_Be\_StringEmpty()

{

Assert.AreEqual(string.Empty, Kata.PrintSpace(0));

}

[TestMethod]

public void PrintSpace\_Input\_1\_Should\_Be\_1Space()

{

Assert.AreEqual(" ", Kata.PrintSpace(1));

}

[TestMethod]

public void PrintSpace\_Input\_2\_Should\_Be\_1Space()

{

Assert.AreEqual(" ", Kata.PrintSpace(2));

}

[TestMethod]

public void Input\_1\_Should\_Be\_1star()

{

CollectionAssert.AreEqual(new string[] { "\*" }, Kata.TowerBuilder(1));

}

[TestMethod]

public void Input\_2\_Should\_Be\_1starAnd3Star()

{

CollectionAssert.AreEqual(new string[] { " \* ", "\*\*\*" }, Kata.TowerBuilder(2));

}

[TestMethod]

public void Input\_3\_Should\_Be\_1starAnd3StarAnd5Star()

{

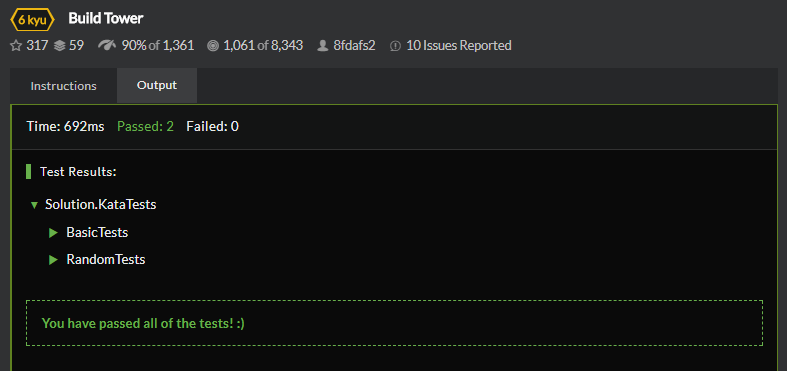
CollectionAssert.AreEqual(new string[] { " \* ", " \*\*\* ", "\*\*\*\*\*" }, Kata.TowerBuilder(3));

}

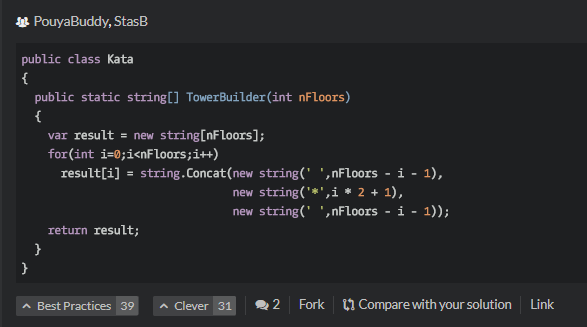
}

```

在Codewars上成功提交了~



提交之後看了一下別人寫的XD

­­­

看起來其他人的寫法也跟我差不多，不過這個比較特別一點，他是用Concat來做的，看起來也是蠻明瞭的~

不得不說以前大學寫這題蠻頭痛的

一次就要完成所有需求是有難度的

像這樣把題目拆解出來，慢慢地完成他

而且還會有測試案例守護你

感覺是真的很棒的

不會怕改了A壞了B

Git url :

<https://github.com/SQZ777/Codewars_BuildTower>

Codewars Link:

<https://www.codewars.com/kata/build-tower/train/csharp>

下一題，明天見!