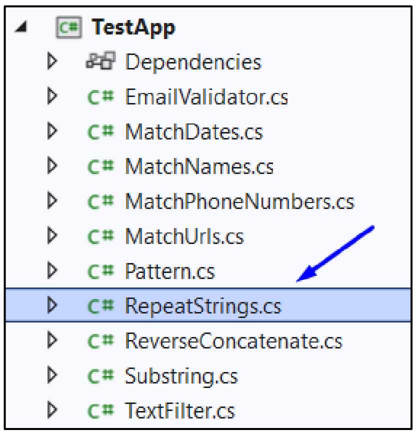
Unit Testing Exercise: Strings and Regular Expressions

Test your tasks in the Judge system: https://judge.softuni.org/Contests/4464

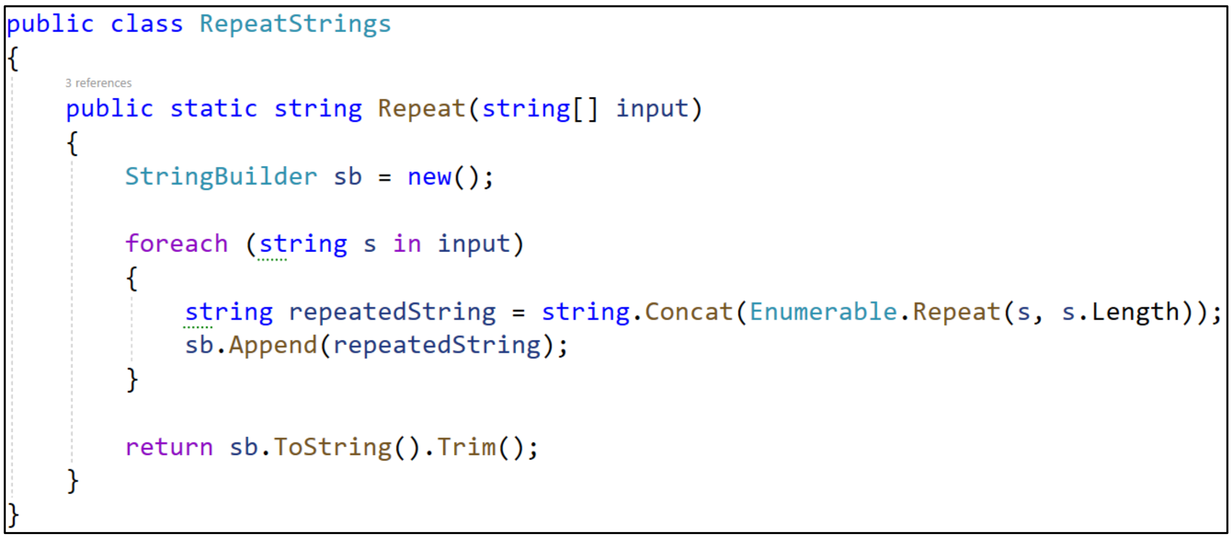
1. Unit Test String Method: Repeat String

Look at the **provided skeleton** and examine the **`RepeatStrings.cs`** class that you will test:



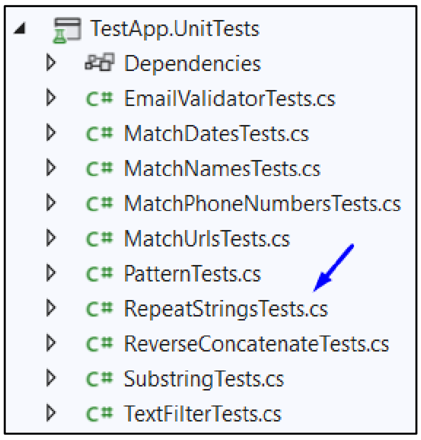
```
TestApp
  Dependencies
  EmailValidator.cs
  MatchDates.cs
  MatchNames.cs
  MatchPhoneNumbers.cs
  MatchUrls.cs
  Pattern.cs
  RepeatStrings.cs
  ReverseConcatenate.cs
  Substring.cs
  TextFilter.cs
```

The method takes in an **array of strings**, and for every string **it repeats it length of the word times** for example the word "**hello**" would be repeated **5 times** because it has **5 letters**:

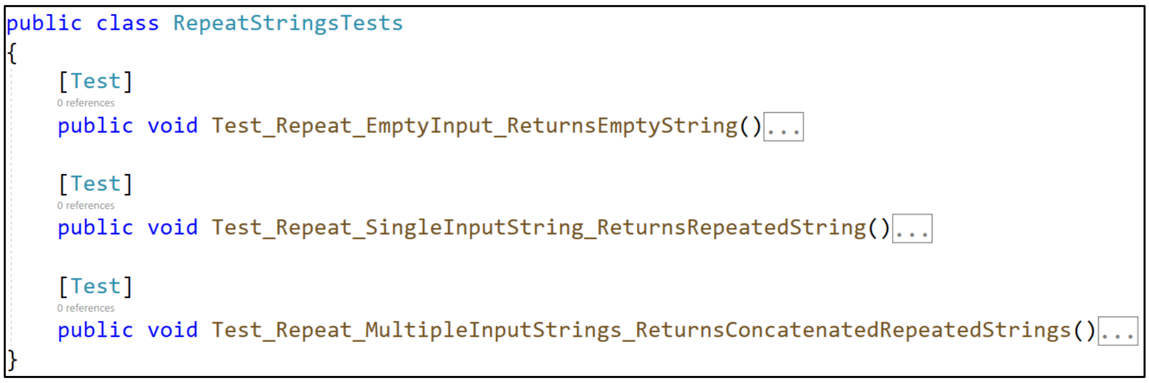


```
public class RepeatStrings
{
    3 references
    public static string Repeat(string[] input)
    {
        StringBuilder sb = new();

        foreach (string s in input)
        {
            string repeatedString = string.Concat(Enumerable.Repeat(s, s.Length));
            sb.Append(repeatedString);
        }

        return sb.ToString().Trim();
    }
}
```

Then, look at the tests inside the **`RepeatStringsTests.cs`** class:

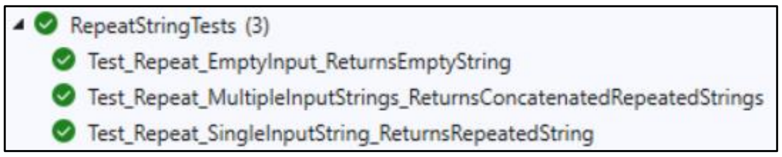


```
TestApp.UnitTests
  Dependencies
  EmailValidatorTests.cs
  MatchDatesTests.cs
  MatchNamesTests.cs
  MatchPhoneNumbersTests.cs
  MatchUrlsTests.cs
  PatternTests.cs
  RepeatStringsTests.cs
  ReverseConcatenateTests.cs
  SubstringTests.cs
  TextFilterTests.cs
```



```
public class RepeatStringsTests
{
    [Test]
    0 references
    public void Test_Repeat_EmptyInput_ReturnsEmptyString()...

    [Test]
    0 references
    public void Test_Repeat_SingleInputString_ReturnsRepeatedString()...

    [Test]
    0 references
    public void Test_Repeat_MultipleInputStrings_ReturnsConcatenatedRepeatedStrings()...
}
```

The first test if **finished** so you have a **reference**, the rest of the tests are **empty,** and your task is to finish them. The tests should run when you're finished:

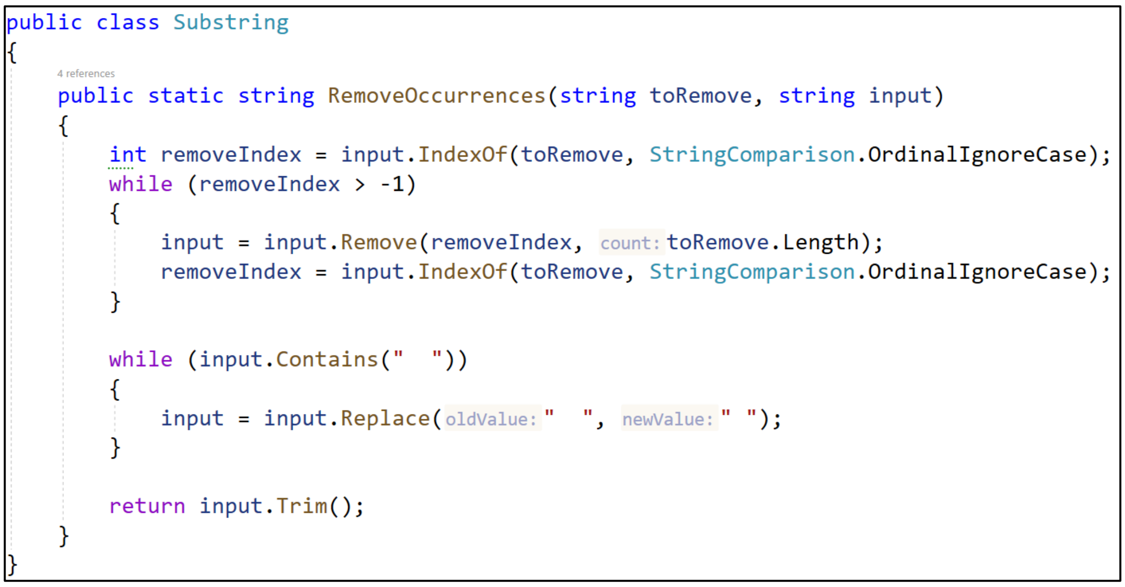


```
◢ ✓ RepeatStringTests (3)
    ✓ Test_Repeat_EmptyInput_ReturnsEmptyString
    ✓ Test_Repeat_MultipleInputStrings_ReturnsConcatenatedRepeatedStrings
    ✓ Test_Repeat_SingleInputString_ReturnsRepeatedString
```

2. Unit Test String Method: Substring

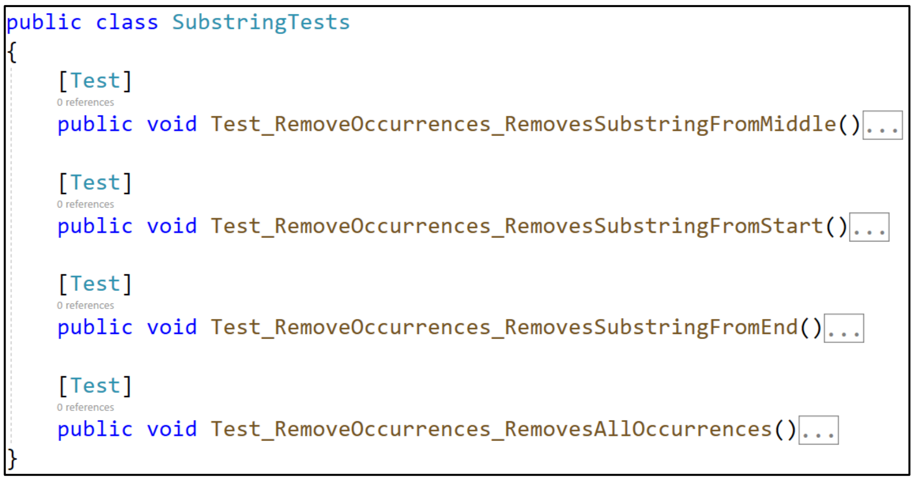
Test a given method which takes in a **string to be removed and a string as text from which the string to be removed**.

The method is found in the **`Substring.cs`** file:



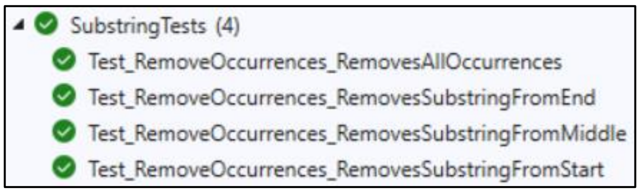
```
public class Substring
{
    4 references
    public static string RemoveOccurrences(string toRemove, string input)
    {
        int removeIndex = input.IndexOf(toRemove, StringComparison.OrdinalIgnoreCase);
        while (removeIndex > -1)
        {
            input = input.Remove(removeIndex, count: toRemove.Length);
            removeIndex = input.IndexOf(toRemove, StringComparison.OrdinalIgnoreCase);
        }

        while (input.Contains("  "))
        {
            input = input.Replace(oldValue: "  ", newValue: " ");
        }

        return input.Trim();
    }
}
```

You are given a **test file `SubstringTests.cs`** which contains **4 tests**. **One** of them has been **finished partially**, and **three** are **empty** for you to finish:



```
public class SubstringTests
{
    [Test]
    0 references
    public void Test_RemoveOccurrences_RemovesSubstringFromMiddle()...

    [Test]
    0 references
    public void Test_RemoveOccurrences_RemovesSubstringFromStart()...

    [Test]
    0 references
    public void Test_RemoveOccurrences_RemovesSubstringFromEnd()...

    [Test]
    0 references
    public void Test_RemoveOccurrences_RemovesAllOccurrences()...
}
```

When you are ready make sure your **tests run:**

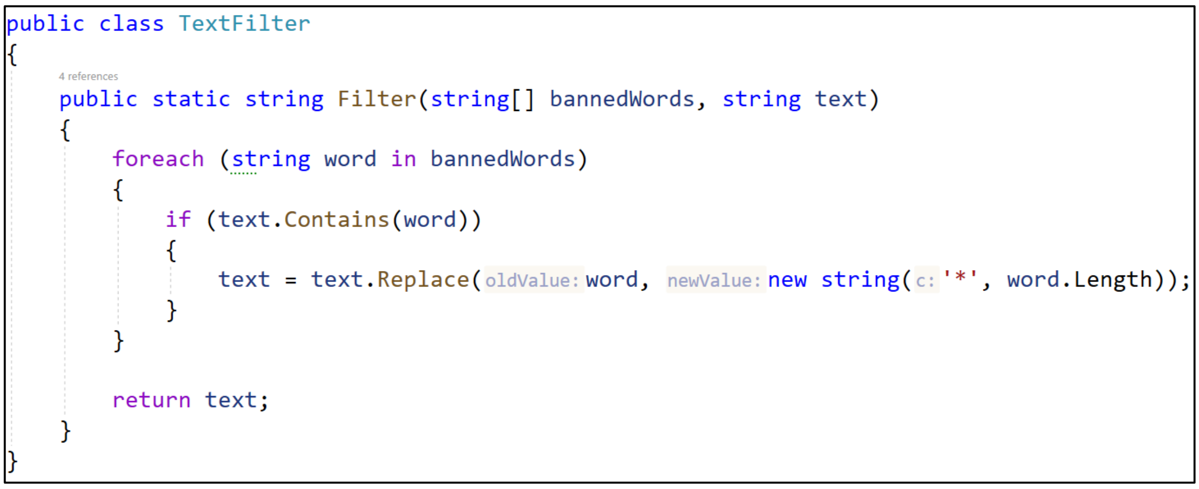


```
◢ ✓ SubstringTests (4)
    ✓ Test_RemoveOccurrences_RemovesAllOccurrences
    ✓ Test_RemoveOccurrences_RemovesSubstringFromEnd
    ✓ Test_RemoveOccurrences_RemovesSubstringFromMiddle
    ✓ Test_RemoveOccurrences_RemovesSubstringFromStart
```

3. Unit Test String Method: Text Filter

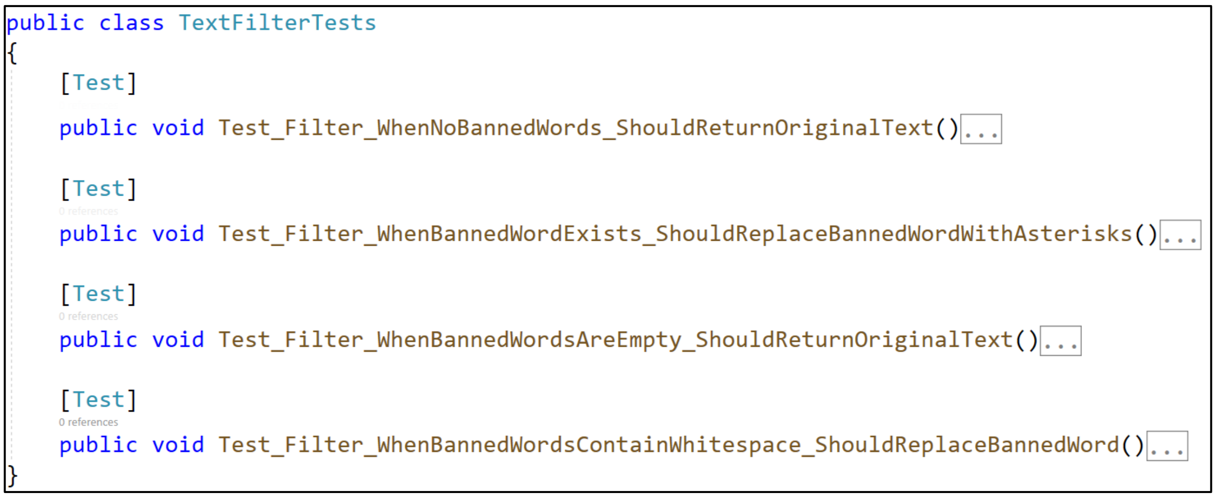
Test a given method which takes in an **array of strings representing banned words and a string representing text** and **blurs out** every **banned word** found by **replacing it** with **asterisks**.

The method is found in the **`TextFilter.cs`** file:



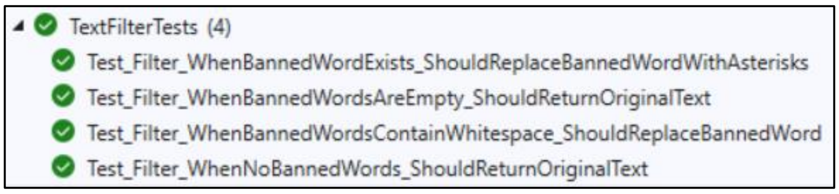
```
public class TextFilter
{
    4 references
    public static string Filter(string[] bannedWords, string text)
    {
        foreach (string word in bannedWords)
        {
            if (text.Contains(word))
            {
                text = text.Replace(oldValue: word, newValue: new string(c: '*', word.Length));
            }
        }

        return text;
    }
}
```

You are given a **test file `TextFilterTests.cs`** which contains **4 tests**. **One** of them has been **finished partially**, and **three** are **empty** for you to finish:



```
public class TextFilterTests
{
    [Test]
    public void Test_Filter_WhenNoBannedWords_ShouldReturnOriginalText()...

    [Test]
    0 references
    public void Test_Filter_WhenBannedWordExists_ShouldReplaceBannedWordWithAsterisks()...

    [Test]
    0 references
    public void Test_Filter_WhenBannedWordsAreEmpty_ShouldReturnOriginalText()...

    [Test]
    0 references
    public void Test_Filter_WhenBannedWordsContainWhitespace_ShouldReplaceBannedWord()...
}
```

When you are ready make sure your **tests run:**

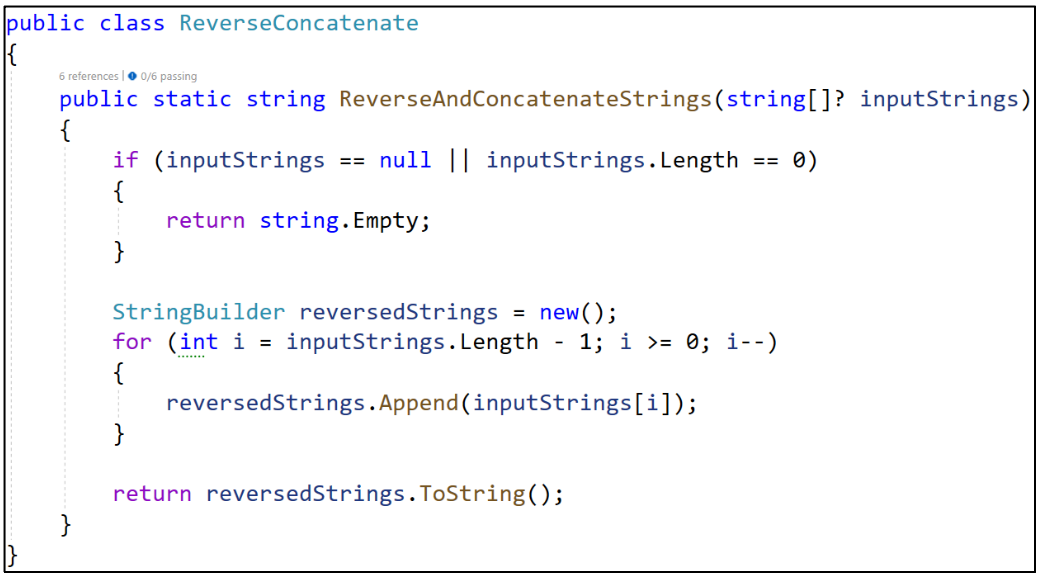


```
TextFilterTests (4)
    Test_Filter_WhenBannedWordExists_ShouldReplaceBannedWordWithAsterisks
    Test_Filter_WhenBannedWordsAreEmpty_ShouldReturnOriginalText
    Test_Filter_WhenBannedWordsContainWhitespace_ShouldReplaceBannedWord
    Test_Filter_WhenNoBannedWords_ShouldReturnOriginalText
```

4. Unit Test String Method: Reverse and Concatenate

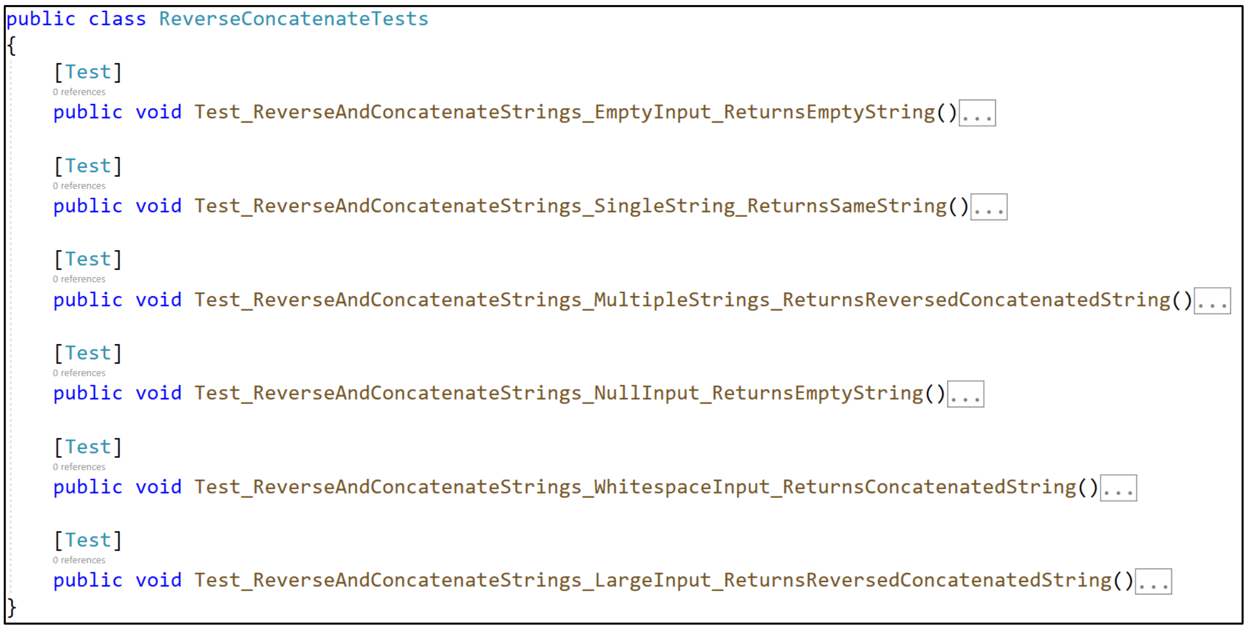
Test a given method which takes in an **array of strings** puts the **words** in a **reverse order** and **concatenates them together**.

The method is found in the **`ReverseConcatenate.cs`** file:



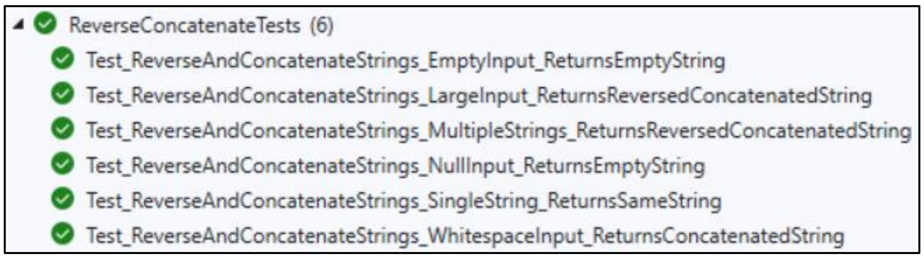
```
public class ReverseConcatenate
{
    6 references | 0/6 passing
    public static string ReverseAndConcatenateStrings(string[]? inputStrings)
    {
        if (inputStrings == null || inputStrings.Length == 0)
        {
            return string.Empty;
        }

        StringBuilder reversedStrings = new();
        for (int i = inputStrings.Length - 1; i >= 0; i--)
        {
            reversedStrings.Append(inputStrings[i]);
        }

        return reversedStrings.ToString();
    }
}
```

You are given a **test file `ReverseConcatenateTests.cs`** which contains **6 tests**. **Two** of them has been **finished partially**, and **four** are **empty** for you to finish:



```
public class ReverseConcatenateTests
{
    [Test]
    0 references
    public void Test_ReverseAndConcatenateStrings_EmptyInput_ReturnsEmptyString()...

    [Test]
    0 references
    public void Test_ReverseAndConcatenateStrings_SingleString_ReturnsSameString()...

    [Test]
    0 references
    public void Test_ReverseAndConcatenateStrings_MultipleStrings_ReturnsReversedConcatenatedString()...

    [Test]
    0 references
    public void Test_ReverseAndConcatenateStrings_NullInput_ReturnsEmptyString()...

    [Test]
    0 references
    public void Test_ReverseAndConcatenateStrings_WhitespaceInput_ReturnsConcatenatedString()...

    [Test]
    0 references
    public void Test_ReverseAndConcatenateStrings_LargeInput_ReturnsReversedConcatenatedString()...
}
```

When you are ready make sure your **tests run:**

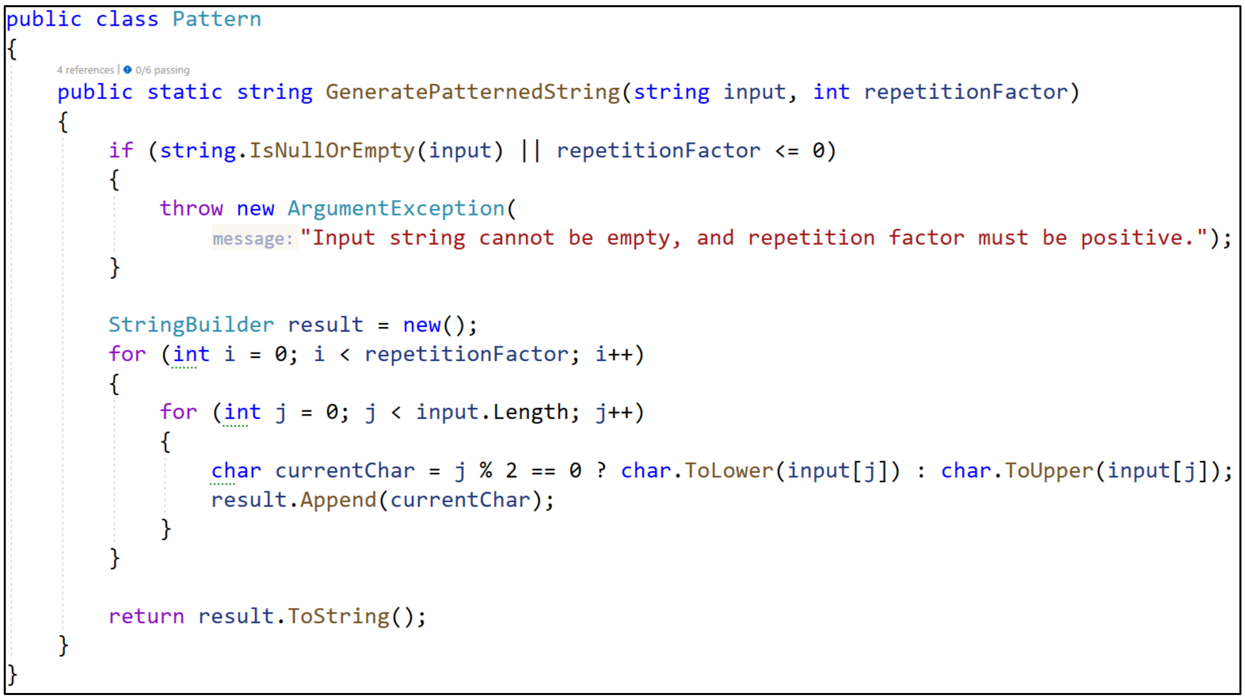


```
◢ ✓ ReverseConcatenateTests (6)
    ✓ Test_ReverseAndConcatenateStrings_EmptyInput_ReturnsEmptyString
    ✓ Test_ReverseAndConcatenateStrings_LargeInput_ReturnsReversedConcatenatedString
    ✓ Test_ReverseAndConcatenateStrings_MultipleStrings_ReturnsReversedConcatenatedString
    ✓ Test_ReverseAndConcatenateStrings_NullInput_ReturnsEmptyString
    ✓ Test_ReverseAndConcatenateStrings_SingleString_ReturnsSameString
    ✓ Test_ReverseAndConcatenateStrings_WhitespaceInput_ReturnsConcatenatedString
```

5. Unit Test String Method: Pattern

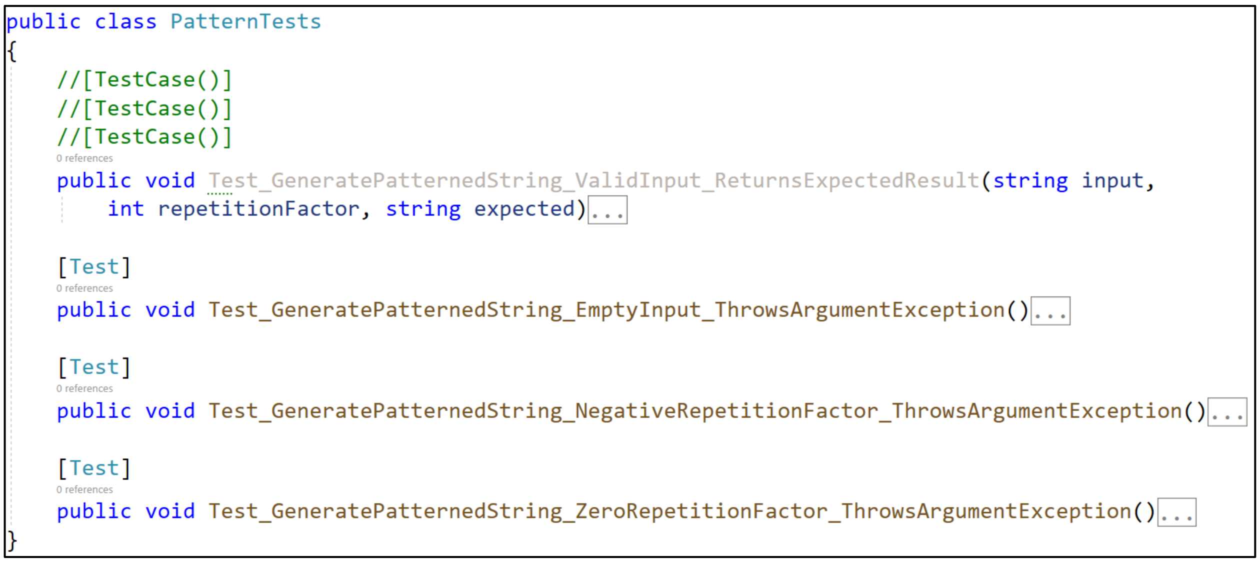
Test a given method which takes in a **string and a number indicating repetition count** then turning every **even letter** to **lowercase** and every **odd letter** to **uppercase** and **repeats** this process as much times as **specified**.

The method is found in the **`Pattern.cs`** file:



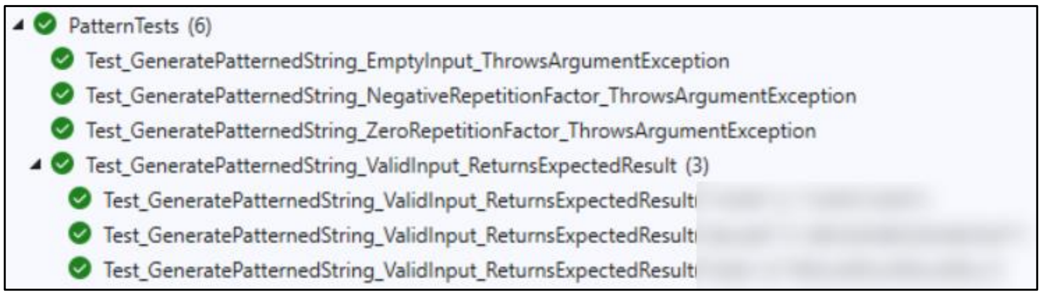
```
public class Pattern
{
    4 references | 0/6 passing
    public static string GeneratePatternedString(string input, int repetitionFactor)
    {
        if (string.IsNullOrEmpty(input) || repetitionFactor <= 0)
        {
            throw new ArgumentException(
                message: "Input string cannot be empty, and repetition factor must be positive.");
        }

        StringBuilder result = new();
        for (int i = 0; i < repetitionFactor; i++)
        {
            for (int j = 0; j < input.Length; j++)
            {
                char currentChar = j % 2 == 0 ? char.ToLower(input[j]) : char.ToUpper(input[j]);
                result.Append(currentChar);
            }
        }

        return result.ToString();
    }
}
```

You are given a **test file `PatternTests.cs`** which contains **6 tests**. **One** of them has been **finished partially**, and **five** are **empty** for you to finish:



```
public class PatternTests
{
    //[TestCase()]
    //[TestCase()]
    //[TestCase()]
    0 references
    public void Test_GeneratePatternedString_ValidInput_ReturnsExpectedResult(string input,
        int repetitionFactor, string expected)...

    [Test]
    0 references
    public void Test_GeneratePatternedString_EmptyInput_ThrowsArgumentException()...

    [Test]
    0 references
    public void Test_GeneratePatternedString_NegativeRepetitionFactor_ThrowsArgumentException()...

    [Test]
    0 references
    public void Test_GeneratePatternedString_ZeroRepetitionFactor_ThrowsArgumentException()...
}
```

When you are ready make sure your **tests run:**

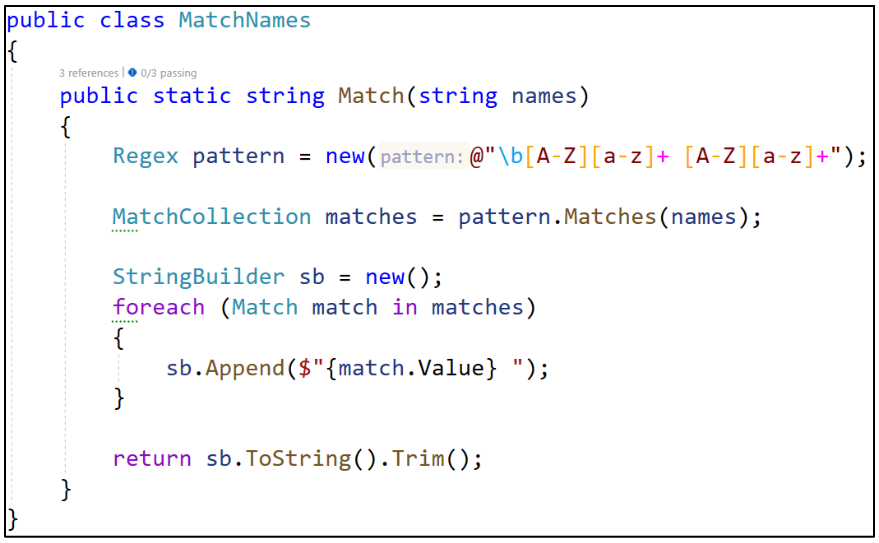


```
◢ ✓ PatternTests (6)
    ✓ Test_GeneratePatternedString_EmptyInput_ThrowsArgumentException
    ✓ Test_GeneratePatternedString_NegativeRepetitionFactor_ThrowsArgumentException
    ✓ Test_GeneratePatternedString_ZeroRepetitionFactor_ThrowsArgumentException
  ◢ ✓ Test_GeneratePatternedString_ValidInput_ReturnsExpectedResult (3)
      ✓ Test_GeneratePatternedString_ValidInput_ReturnsExpectedResult[redacted]
      ✓ Test_GeneratePatternedString_ValidInput_ReturnsExpectedResult[redacted]
      ✓ Test_GeneratePatternedString_ValidInput_ReturnsExpectedResult[redacted]
```

6. Unit Test Regular Expression: Match Names

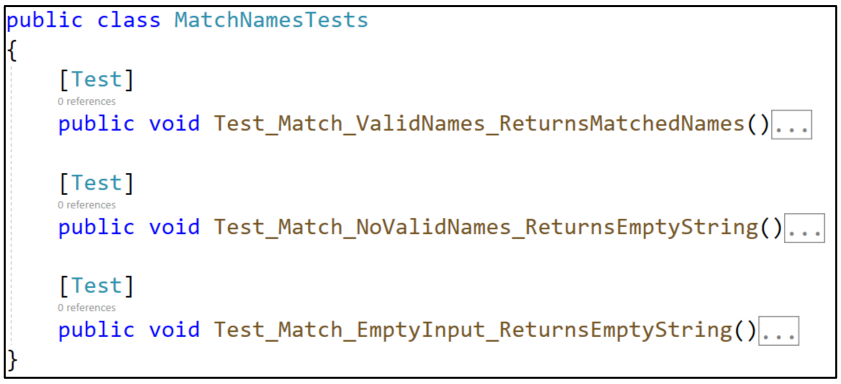
Test a given method which takes in a **string array of names** and matches names in the form of **`Firstname Lastname`** starting both with **capital letters**.

The method is found in the **`MatchNames.cs`** file:



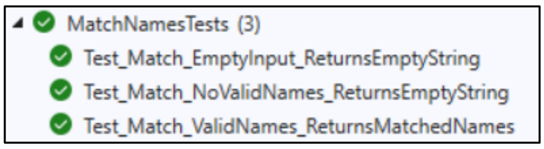
```
public class MatchNames
{
    3 references | 0/3 passing
    public static string Match(string names)
    {
        Regex pattern = new(pattern: @"\b[A-Z][a-z]+ [A-Z][a-z]+");

        MatchCollection matches = pattern.Matches(names);

        StringBuilder sb = new();
        foreach (Match match in matches)
        {
            sb.Append($"{match.Value} ");
        }

        return sb.ToString().Trim();
    }
}
```

You are given a **test file `MatchNamesTests.cs`** which contains **3 tests**. **One** of them has been **finished**, and **two** are **empty** for you to finish:



```
public class MatchNamesTests
{
    [Test]
    0 references
    public void Test_Match_ValidNames_ReturnsMatchedNames()...

    [Test]
    0 references
    public void Test_Match_NoValidNames_ReturnsEmptyString()...

    [Test]
    0 references
    public void Test_Match_EmptyInput_ReturnsEmptyString()...
}
```

When you are ready make sure your **tests run:**

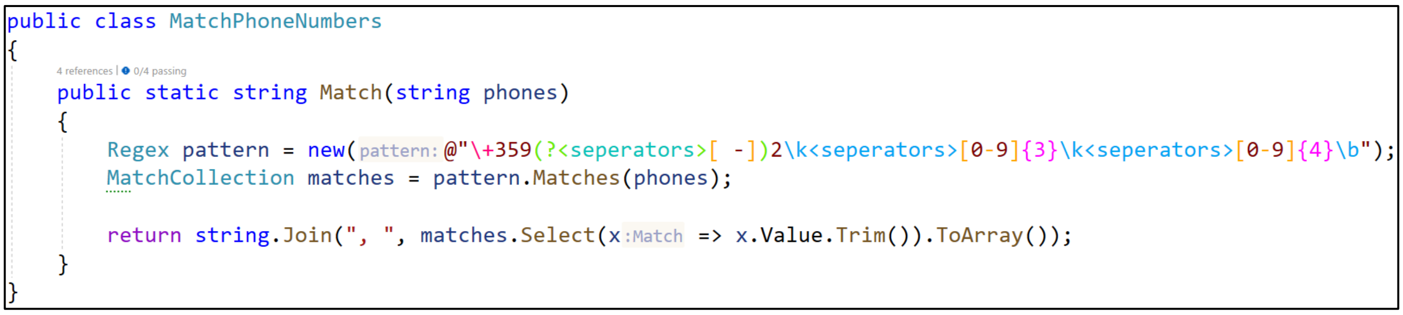


```
◢ ✓ MatchNamesTests (3)
    ✓ Test_Match_EmptyInput_ReturnsEmptyString
    ✓ Test_Match_NoValidNames_ReturnsEmptyString
    ✓ Test_Match_ValidNames_ReturnsMatchedNames
```

7. Unit Test Regular Expression: Match Phone Numbers

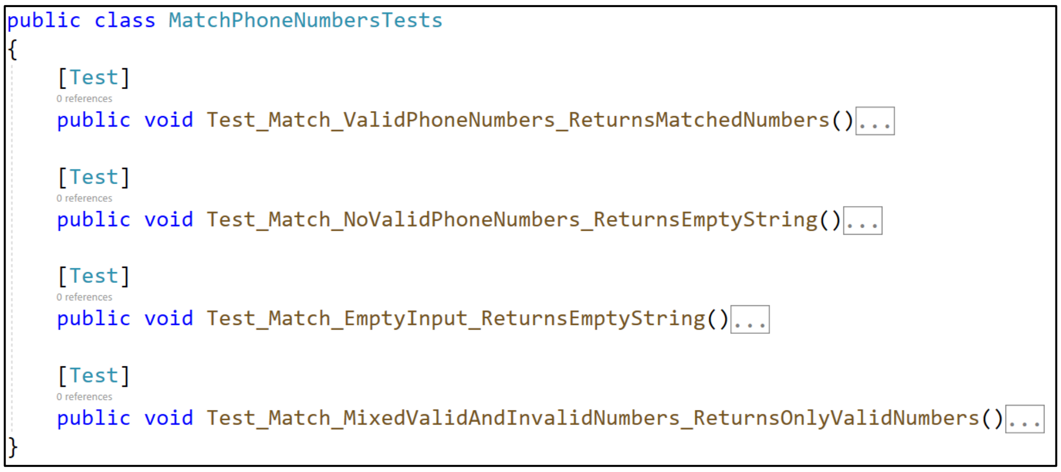
Test a given method which takes in a **string array of phone numbers** and matches phones in the form of **+359** followed by **either** a **space** or a **hyphen**, then the **area code '2,'** followed by **three digits**, and finally, **four more digits** at the end.

The method is found in the **`MatchPhoneNumbers.cs`** file:



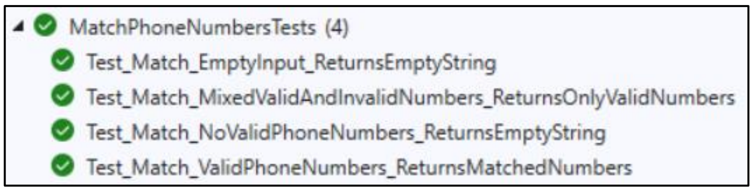
```
public class MatchPhoneNumbers
{
    4 references | 0/4 passing
    public static string Match(string phones)
    {
        Regex pattern = new(pattern: @"\+359(?<seperators>[ -])2\k<seperators>[0-9]{3}\k<seperators>[0-9]{4}\b");
        MatchCollection matches = pattern.Matches(phones);

        return string.Join(", ", matches.Select(x:Match => x.Value.Trim()).ToArray());
    }
}
```

You are given a **test file `MatchPhoneNumbersTests.cs`** which contains **4 tests**. **One** of them has been **finished partially**, and **three** are **empty** for you to finish:



```
public class MatchPhoneNumbersTests
{
    [Test]
    0 references
    public void Test_Match_ValidPhoneNumbers_ReturnsMatchedNumbers()...

    [Test]
    0 references
    public void Test_Match_NoValidPhoneNumbers_ReturnsEmptyString()...

    [Test]
    0 references
    public void Test_Match_EmptyInput_ReturnsEmptyString()...

    [Test]
    0 references
    public void Test_Match_MixedValidAndInvalidNumbers_ReturnsOnlyValidNumbers()...
}
```

When you are ready make sure your **tests run:**

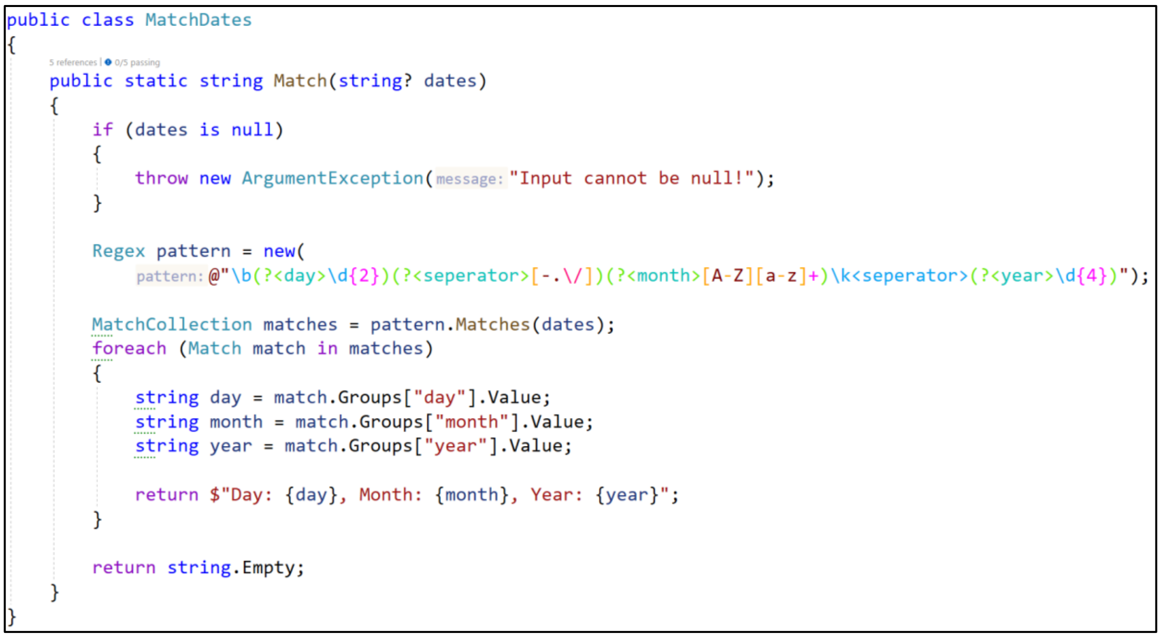


```
◢ ✓ MatchPhoneNumbersTests (4)
    ✓ Test_Match_EmptyInput_ReturnsEmptyString
    ✓ Test_Match_MixedValidAndInvalidNumbers_ReturnsOnlyValidNumbers
    ✓ Test_Match_NoValidPhoneNumbers_ReturnsEmptyString
    ✓ Test_Match_ValidPhoneNumbers_ReturnsMatchedNumbers
```

8. Unit Test Regular Expression: Match Dates

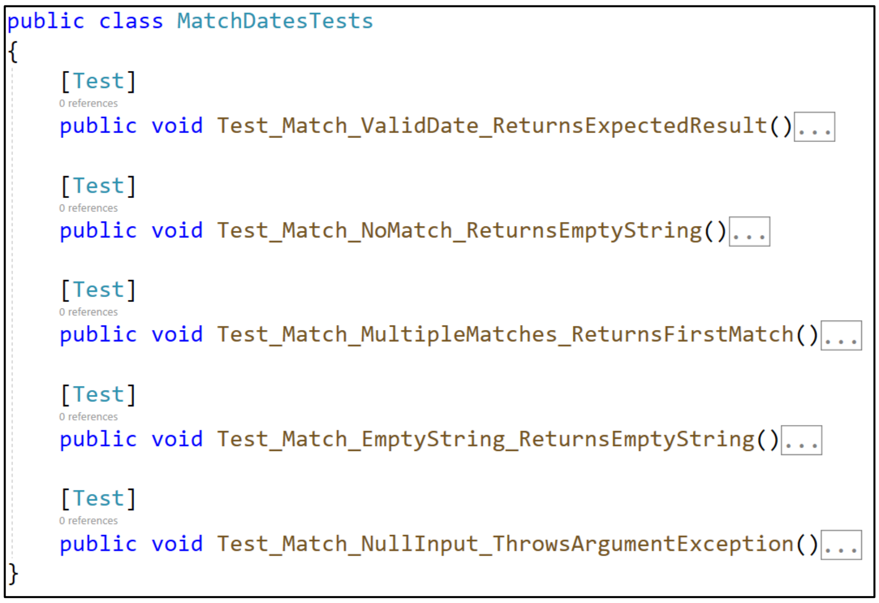
Test a given method which takes in a **string array of dates** and matches dates in the form of **two digits** for the **day**, **separated** by a **dash**, **period**, **forward slash**, or **backslash**, followed by the **abbreviated month name** and another occurrence of the same **separator**, ending with a **four-digit year**.

The method is found in the `MatchDates.cs` file:



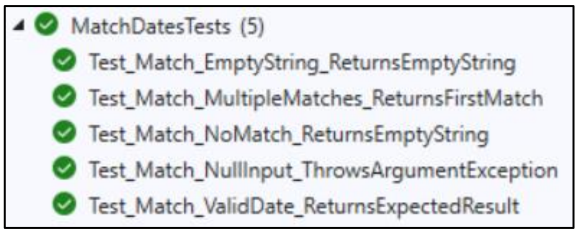
```
public class MatchDates
{
    5 references | ● 0/5 passing
    public static string Match(string? dates)
    {
        if (dates is null)
        {
            throw new ArgumentException(message: "Input cannot be null!");
        }

        Regex pattern = new(
            pattern: @"\b(?<day>\d{2})(?<seperator>[-.\/])(?<month>[A-Z][a-z]+)\k<seperator>(?<year>\d{4})");

        MatchCollection matches = pattern.Matches(dates);
        foreach (Match match in matches)
        {
            string day = match.Groups["day"].Value;
            string month = match.Groups["month"].Value;
            string year = match.Groups["year"].Value;

            return $"Day: {day}, Month: {month}, Year: {year}";
        }

        return string.Empty;
    }
}
```

You are given a **test file `MatchDatesTests.cs`** which contains **5 tests**. **Two** of them has been **finished partially**, and **three** are **empty** for you to finish:



```
public class MatchDatesTests
{
    [Test]
    0 references
    public void Test_Match_ValidDate_ReturnsExpectedResult()...

    [Test]
    0 references
    public void Test_Match_NoMatch_ReturnsEmptyString()...

    [Test]
    0 references
    public void Test_Match_MultipleMatches_ReturnsFirstMatch()...

    [Test]
    0 references
    public void Test_Match_EmptyString_ReturnsEmptyString()...

    [Test]
    0 references
    public void Test_Match_NullInput_ThrowsArgumentException()...
}
```

When you are ready make sure your **tests run:**

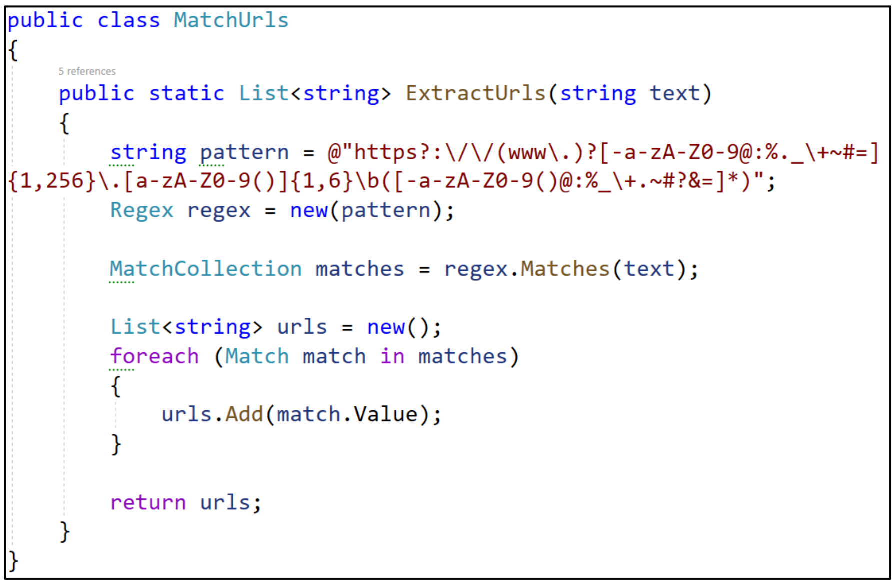


```
◢ MatchDatesTests (5)
    Test_Match_EmptyString_ReturnsEmptyString
    Test_Match_MultipleMatches_ReturnsFirstMatch
    Test_Match_NoMatch_ReturnsEmptyString
    Test_Match_NullInput_ThrowsArgumentException
    Test_Match_ValidDate_ReturnsExpectedResult
```

9. Unit Test Regular Expression: Match URLs

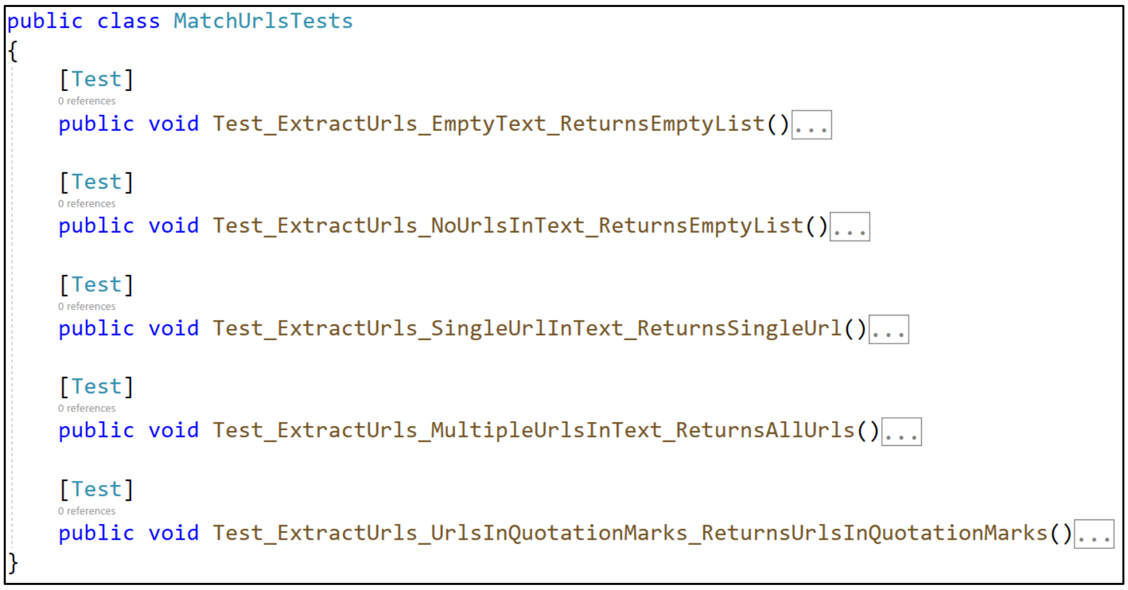
Test a given method which takes in a **string array of URLs** and matches them in the standard **HTTP** or **HTTPS format**, **optionally** preceded by **'`www.`'**, and consisting of **valid characters** for **domain names** and **query parameters**.

The method is found in the **`MatchUrls.cs`** file:



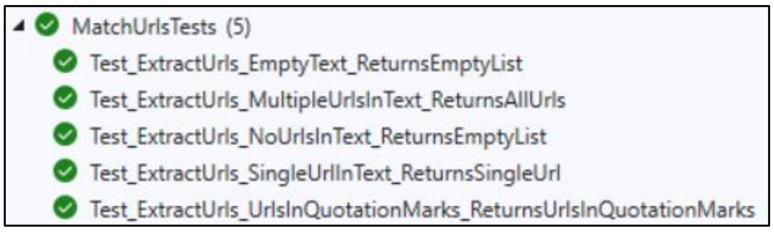
```
public class MatchUrls
{
    5 references
    public static List<string> ExtractUrls(string text)
    {
        string pattern = @"https?:\/\/(www\.)?[-a-zA-Z0-9@:%._\+~#=]
{1,256}\.[a-zA-Z0-9()]{1,6}\b([-a-zA-Z0-9()@:%_\+.~#?&=]*)";
        Regex regex = new(pattern);

        MatchCollection matches = regex.Matches(text);

        List<string> urls = new();
        foreach (Match match in matches)
        {
            urls.Add(match.Value);
        }

        return urls;
    }
}
```

You are given a **test file `MatchUrlsTests.cs`** which contains **5 tests**. **Two** of them has been **finished partially**, and **three** are **empty** for you to finish:



```
public class MatchUrlsTests
{
    [Test]
    0 references
    public void Test_ExtractUrls_EmptyText_ReturnsEmptyList()...

    [Test]
    0 references
    public void Test_ExtractUrls_NoUrlsInText_ReturnsEmptyList()...

    [Test]
    0 references
    public void Test_ExtractUrls_SingleUrlInText_ReturnsSingleUrl()...

    [Test]
    0 references
    public void Test_ExtractUrls_MultipleUrlsInText_ReturnsAllUrls()...

    [Test]
    0 references
    public void Test_ExtractUrls_UrlsInQuotationMarks_ReturnsUrlsInQuotationMarks()...
}
```

When you are ready make sure your **tests run:**

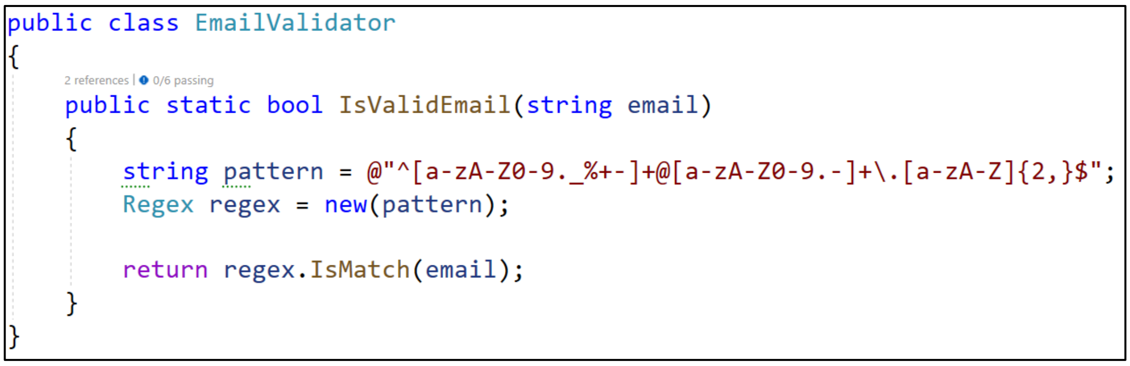


```
◢ ✓ MatchUrlsTests (5)
    ✓ Test_ExtractUrls_EmptyText_ReturnsEmptyList
    ✓ Test_ExtractUrls_MultipleUrlsInText_ReturnsAllUrls
    ✓ Test_ExtractUrls_NoUrlsInText_ReturnsEmptyList
    ✓ Test_ExtractUrls_SingleUrlInText_ReturnsSingleUrl
    ✓ Test_ExtractUrls_UrlsInQuotationMarks_ReturnsUrlsInQuotationMarks
```

10. Unit Test Regular Expression: Email Validator

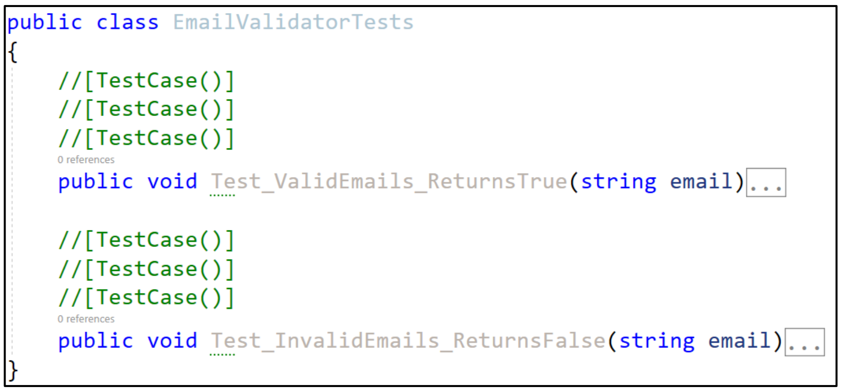
Test a given method which takes in a **string array of emails** and matches **valid emails** which are **combination** of **letters**, **numbers**, **dots**, **underscores**, **percentage signs**, **plus signs**, or **hyphens** before the '**@**' **symbol**, followed by a **domain** containing **letters**, **numbers**, **hyphens**, and **dots**, and **ending** with a **top-level domain** of at least **two letters**.

The method is found in the **`EmailValidator.cs`** file:



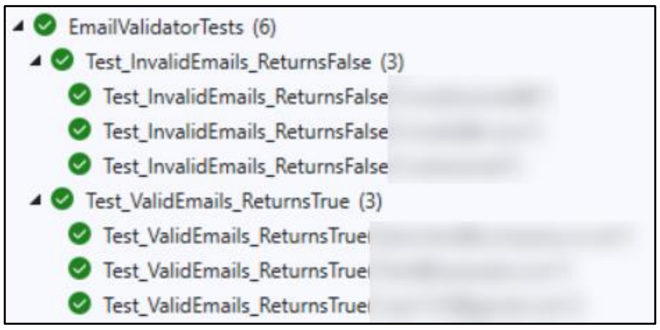
```
public class EmailValidator
{
    2 references | ● 0/6 passing
    public static bool IsValidEmail(string email)
    {
        string pattern = @"^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$";
        Regex regex = new(pattern);

        return regex.IsMatch(email);
    }
}
```

You are given a **test file `EmailValidatorTests.cs`** which contains **6 test cases for you to write**:



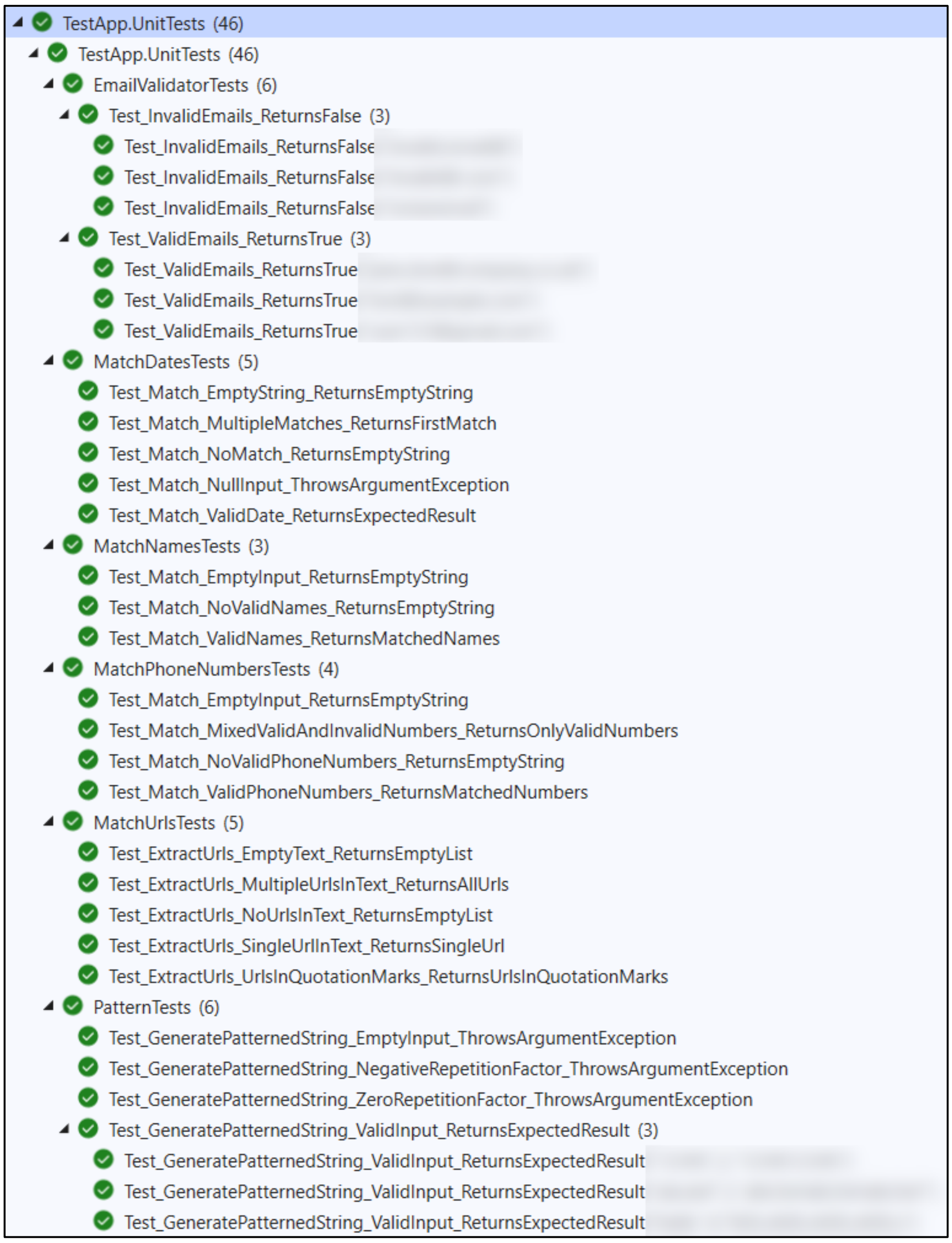
```
public class EmailValidatorTests
{
    //[TestCase()]
    //[TestCase()]
    //[TestCase()]
    0 references
    public void Test_ValidEmails_ReturnsTrue(string email)...

    //[TestCase()]
    //[TestCase()]
    //[TestCase()]
    0 references
    public void Test_InvalidEmails_ReturnsFalse(string email)...
}
```

When you are ready make sure your **tests run:**

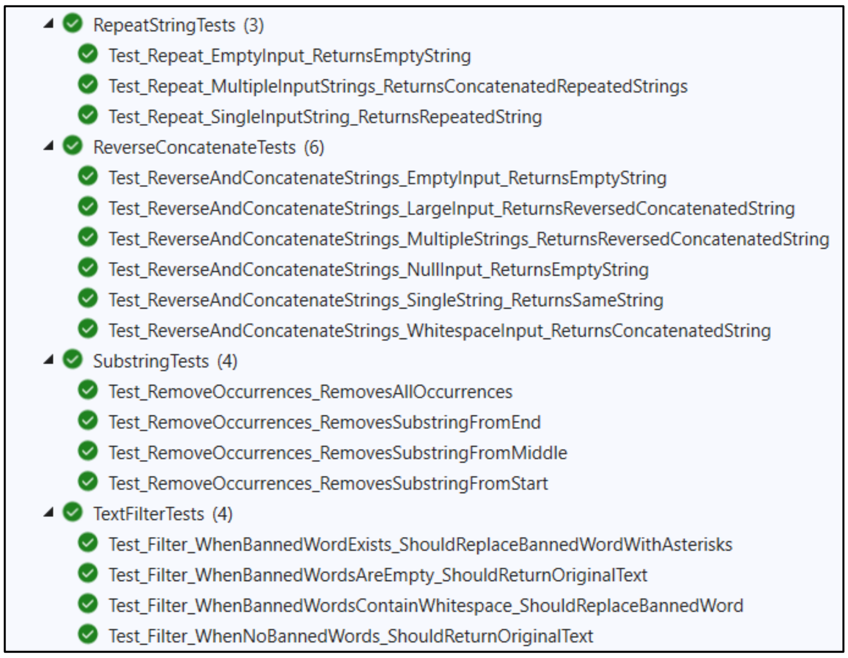


```
◢ ✓ EmailValidatorTests (6)
  ◢ ✓ Test_InvalidEmails_ReturnsFalse (3)
      ✓ Test_InvalidEmails_ReturnsFalse
      ✓ Test_InvalidEmails_ReturnsFalse
      ✓ Test_InvalidEmails_ReturnsFalse
  ◢ ✓ Test_ValidEmails_ReturnsTrue (3)
      ✓ Test_ValidEmails_ReturnsTrue
      ✓ Test_ValidEmails_ReturnsTrue
      ✓ Test_ValidEmails_ReturnsTrue
```

Finally make sure **all** tests run:



```
◢ ✓ TestApp.UnitTests (46)
  ◢ ✓ TestApp.UnitTests (46)
    ◢ ✓ EmailValidatorTests (6)
      ◢ ✓ Test_InvalidEmails_ReturnsFalse (3)
          ✓ Test_InvalidEmails_ReturnsFalse
          ✓ Test_InvalidEmails_ReturnsFalse
          ✓ Test_InvalidEmails_ReturnsFalse
      ◢ ✓ Test_ValidEmails_ReturnsTrue (3)
          ✓ Test_ValidEmails_ReturnsTrue
          ✓ Test_ValidEmails_ReturnsTrue
          ✓ Test_ValidEmails_ReturnsTrue
    ◢ ✓ MatchDatesTests (5)
        ✓ Test_Match_EmptyString_ReturnsEmptyString
        ✓ Test_Match_MultipleMatches_ReturnsFirstMatch
        ✓ Test_Match_NoMatch_ReturnsEmptyString
        ✓ Test_Match_NullInput_ThrowsArgumentException
        ✓ Test_Match_ValidDate_ReturnsExpectedResult
    ◢ ✓ MatchNamesTests (3)
        ✓ Test_Match_EmptyInput_ReturnsEmptyString
        ✓ Test_Match_NoValidNames_ReturnsEmptyString
        ✓ Test_Match_ValidNames_ReturnsMatchedNames
    ◢ ✓ MatchPhoneNumbersTests (4)
        ✓ Test_Match_EmptyInput_ReturnsEmptyString
        ✓ Test_Match_MixedValidAndInvalidNumbers_ReturnsOnlyValidNumbers
        ✓ Test_Match_NoValidPhoneNumbers_ReturnsEmptyString
        ✓ Test_Match_ValidPhoneNumbers_ReturnsMatchedNumbers
    ◢ ✓ MatchUrlsTests (5)
        ✓ Test_ExtractUrls_EmptyText_ReturnsEmptyList
        ✓ Test_ExtractUrls_MultipleUrlsInText_ReturnsAllUrls
        ✓ Test_ExtractUrls_NoUrlsInText_ReturnsEmptyList
        ✓ Test_ExtractUrls_SingleUrlInText_ReturnsSingleUrl
        ✓ Test_ExtractUrls_UrlsInQuotationMarks_ReturnsUrlsInQuotationMarks
    ◢ ✓ PatternTests (6)
        ✓ Test_GeneratePatternedString_EmptyInput_ThrowsArgumentException
        ✓ Test_GeneratePatternedString_NegativeRepetitionFactor_ThrowsArgumentException
        ✓ Test_GeneratePatternedString_ZeroRepetitionFactor_ThrowsArgumentException
      ◢ ✓ Test_GeneratePatternedString_ValidInput_ReturnsExpectedResult (3)
          ✓ Test_GeneratePatternedString_ValidInput_ReturnsExpectedResult
          ✓ Test_GeneratePatternedString_ValidInput_ReturnsExpectedResult
          ✓ Test_GeneratePatternedString_ValidInput_ReturnsExpectedResult
```



```
    ◢ ✓ RepeatStringTests (3)
        ✓ Test_Repeat_EmptyInput_ReturnsEmptyString
        ✓ Test_Repeat_MultipleInputStrings_ReturnsConcatenatedRepeatedStrings
        ✓ Test_Repeat_SingleInputString_ReturnsRepeatedString
    ◢ ✓ ReverseConcatenateTests (6)
        ✓ Test_ReverseAndConcatenateStrings_EmptyInput_ReturnsEmptyString
        ✓ Test_ReverseAndConcatenateStrings_LargeInput_ReturnsReversedConcatenatedString
        ✓ Test_ReverseAndConcatenateStrings_MultipleStrings_ReturnsReversedConcatenatedString
        ✓ Test_ReverseAndConcatenateStrings_NullInput_ReturnsEmptyString
        ✓ Test_ReverseAndConcatenateStrings_SingleString_ReturnsSameString
        ✓ Test_ReverseAndConcatenateStrings_WhitespaceInput_ReturnsConcatenatedString
    ◢ ✓ SubstringTests (4)
        ✓ Test_RemoveOccurrences_RemovesAllOccurrences
        ✓ Test_RemoveOccurrences_RemovesSubstringFromEnd
        ✓ Test_RemoveOccurrences_RemovesSubstringFromMiddle
        ✓ Test_RemoveOccurrences_RemovesSubstringFromStart
    ◢ ✓ TextFilterTests (4)
        ✓ Test_Filter_WhenBannedWordExists_ShouldReplaceBannedWordWithAsterisks
        ✓ Test_Filter_WhenBannedWordsAreEmpty_ShouldReturnOriginalText
        ✓ Test_Filter_WhenBannedWordsContainWhitespace_ShouldReplaceBannedWord
        ✓ Test_Filter_WhenNoBannedWords_ShouldReturnOriginalText
```