

Our Solution(s)

Run Code

Your Solutions

Run Code

Solution 1

```
1 // Copyright © 2020 AlgoExpert, LLC. All rights reserved.
2
3 using System.Text;
4 using System.Collections.Generic;
5
6 public class Program {
7     // O(n^2 + m) time | O(n + m) space
8     public static string[] PatternMatcher(string pattern, string str) {
9         if (pattern.Length > str.Length) {
10             return new string[] { };
11         }
12         char[] newPattern = getNewPattern(pattern);
13         bool didSwitch = newPattern[0] != pattern[0];
14         Dictionary<char, int> counts = new Dictionary<char, int>();
15         counts['x'] = 0;
16         counts['y'] = 0;
17         int firstYPos = getCountsAndFirstYPos(newPattern, counts);
18         if (counts['y'] != 0) {
19             for (int lenOfX = 1; lenOfX < str.Length; lenOfX++) {
20                 double lenOfY =
21                     ((double)str.Length - (double)lenOfX *
22                     (double)counts['x']) /
23                     (double)counts['y'];
24                 if (lenOfY <= 0 || lenOfY % 1 != 0) {
25                     continue;
26                 }
27                 int yIdx = firstYPos * lenOfX;
28                 string x = str.Substring(0, lenOfX);
29                 string y = str.Substring(yIdx, (int)lenOfY);
30                 string potentialMatch = buildPotentialMatch(newPattern, x, y);
31                 if (str.Equals(potentialMatch)) {
32                     return didSwitch ? new string[] { y, x } : new string[] { x,
33                                                                                       y };
34                 }
35             }
36         }
37         return new string[] { };
38     }
39
40     private static char[] getNewPattern(string pattern) {
41         char[] newPattern = new char[pattern.Length];
42         for (int i = 0; i < pattern.Length; i++) {
43             newPattern[i] = pattern[i] == 'x' ? 'y' : 'x';
44         }
45         return newPattern;
46     }
47
48     private static int getCountsAndFirstYPos(char[] newPattern, Dictionary<char, int> counts) {
49         int firstYPos = 0;
50         for (int i = 0; i < newPattern.Length; i++) {
51             if (newPattern[i] == 'y') {
52                 firstYPos = i;
53                 counts['y']++;
54             } else {
55                 counts['x']++;
56             }
57         }
58         return firstYPos;
59     }
60
61     private static string buildPotentialMatch(char[] newPattern, string x, string y) {
62         string potentialMatch = new string(newPattern.Length, ' ');
63         for (int i = 0; i < x.Length; i++) {
64             potentialMatch[i] = x[i] == 'x' ? 'y' : 'x';
65         }
66         for (int i = 0; i < y.Length; i++) {
67             potentialMatch[firstYPos + i] = y[i];
68         }
69         return potentialMatch;
70     }
71 }
```

Solution 1

Solution 2

Solution 3

```
1 public class Program {
2     public static string[] PatternMatcher(string pattern, string str) {
3         // Write your code here.
4         return null;
5     }
6 }
7
```

Our Tests

Custom Output

Submit Code

```
1 public class Program {
2     public static string[] PatternMatcher(string pattern, string str) {
3         // Write your code here.
4         return null;
5     }
6 }
```

```
1 public class Program {
2     public static string[] PatternMatcher(string pattern, string str) {
3         // Write your code here.
4         return null;
5     }
6 }
```

```

10     return string, expected
11 }
12
13 func Test(t *testing.T) {
14     string, expected = "100", "100"
15     string, expected = "1000", "1000"
16     string, expected = "10000", "10000"
17     string, expected = "100000", "100000"
18     string, expected = "1000000", "1000000"
19 }
20
21 func Test(t *testing.T) {
22     string, expected = "100", "100"
23 }

```

Run or submit code when you're ready.