

Our Solution(s)		Run Code	Your Solutions			Run Code
-----------------	--	----------	----------------	--	--	----------

Solution 1	Solution 2
<pre>1 // Copyright © 2020 AlgoExpert, LLC. All rights reserved. 2 3 package main 4 5 type substring struct { 6 left int 7 right int 8 } 9 10 func (ss substring) length() int { 11 return ss.right - ss.left 12 } 13 14 // O(n^2) time O(1) space 15 func LongestPalindromicSubstring(str string) string { 16 result := substring{0, 1} 17 for i := 1; i < len(str); i++ { 18 odd := getLongestPalindromeFrom(str, i-1, i+1) 19 even := getLongestPalindromeFrom(str, i-1, i) 20 longest := even 21 if odd.length() > even.length() { 22 longest = odd 23 } 24 if longest.length() > result.length() { 25 result = longest 26 } 27 } 28 return str[result.left:result.right] 29 } 30 31 func getLongestPalindromeFrom(str string, leftIndex, rightIndex 32 for leftIndex >= 0 && rightIndex < len(str) { 33 if str[leftIndex] != str[rightIndex] {</pre>	

Solution 1	Solution 2	Solution 3
<pre>1 package main 2 3 func LongestPalindromicSubstring(str string) string { 4 // Write your code here. 5 return "" 6 } 7</pre>		

```
18 report()
19
20 "*****"
21 }
22
23 Run in RStudio: Testcases: Testcase 2
24 expected = "1"
25 output = longestPalindromeLength("1")
26 report(expected, expected, output)
27 }
28
29 Run in RStudio: Testcases: Testcase 2
30 expected = "121"
31 output = longestPalindromeLength("121")
32 report(expected, expected, output)
33 }
34
35 Run in RStudio: Testcases: Testcase 2
36 expected = "12321"
```

Run or submit code when you're ready.