Our Solution(s) Run Code

Solution 1 Solution 2

```python
# Copyright © 2020 AlgoExpert, LLC. All rights reserved.

# O(n^2) time | O(1) space
def longestPalindromicSubstring(string):
    currentLongest = [0, 1]
    for i in range(1, len(string)):
        odd = getLongestPalindromeFrom(string, i - 1, i + 1)
        even = getLongestPalindromeFrom(string, i - 1, i)
        longest = max(odd, even, key=lambda x: x[1] - x[0])
        currentLongest = max(longest, currentLongest, key=lambda
    return string[currentLongest[0] : currentLongest[1]]


def getLongestPalindromeFrom(string, leftIdx, rightIdx):
    while leftIdx >= 0 and rightIdx < len(string):
        if string[leftIdx] != string[rightIdx]:
            break
        leftIdx -= 1
        rightIdx += 1
    return [leftIdx + 1, rightIdx]

```

Your Solutions Run Code

Solution 1 Solution 2 Solution 3

```python
def longestPalindromicSubstring(string):
    # Write your code here.
    pass

```

Our Tests

Custom Output Submit Code

```
import unittest


class TestSProgram(unittest.TestCase):
    def test_case_1(self):
        self.assertEqual(program.longestPalindrome("a"),
                         "a")

    def test_case_2(self):
        self.assertEqual(program.longestPalindrome("ca"),
                         "c")

    def test_case_3(self):
        self.assertEqual(program.longestPalindrome("cac"),
                         "cac")

    def test_case_4(self):
        self.assertEqual(program.longestPalindrome("cabad"),
                         "aba")

    def test_case_5(self):
        self.assertEqual(program.longestPalindrome("cabbad"),
                         "abba")

    def test_case_6(self):
        self.assertEqual(
```

Run or submit code when you're ready.