Our Solution(s) Run Code

Solution 1 Solution 2

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

# O(n^3) time | O(1) space
def longestPalindromicSubstring(string):
    longest = ""
    for i in range(len(string)):
        for j in range(i, len(string)):
            substring = string[i : j + 1]
            if len(substring) > len(longest) and isPalindrome(su
                longest = substring
    return longest


def isPalindrome(string):
    leftIdx = 0
    rightIdx = len(string) - 1
    while leftIdx < rightIdx:
        if string[leftIdx] != string[rightIdx]:
            return False
        leftIdx += 1
        rightIdx -= 1
    return True

```

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.