Our Solution(s) Run Code

Solution 1 Solution 2

```cpp
// Copyright © 2020 AlgoExpert, LLC. All rights reserved.

#include <vector>
using namespace std;

vector<int> getLongestPalindromeFrom(string str, int leftIdx, int rightIdx);

// O(n^2) time | O(1) space
string longestPalindromicSubstring(string str) {
  vector<int> currentLongest{0, 1};
  for (int i = 1; i < str.length(); i++) {
    vector<int> odd = getLongestPalindromeFrom(str, i - 1, i + 1);
    vector<int> even = getLongestPalindromeFrom(str, i - 1, i);
    vector<int> longest = odd[1] - odd[0] > even[1] - even[0] ? odd : even;
    currentLongest =
        currentLongest[1] - currentLongest[0] > longest[1] - longest[0]
            ? currentLongest
            : longest;
  }
  return str.substr(currentLongest[0], currentLongest[1] - currentLongest[0]);
}

vector<int> getLongestPalindromeFrom(string str, int leftIdx, int rightIdx) {
  while (leftIdx >= 0 && rightIdx < str.length()) {
    if (str[leftIdx] != str[rightIdx]) {
      break;
    }
    leftIdx--;
    rightIdx++;
  }
  return vector<int>{leftIdx + 1, rightIdx};
}

```

Your Solutions Run Code

Solution 1 Solution 2 Solution 3

```cpp
using namespace std;

string longestPalindromicSubstring(string str) {
  // Write your code here.
  return "";
}

```

Custom Output Raw Output Submit Code

Run or submit code when you're ready.