**177. Given a string s, return the longest palindromic substring in S.**

**Example 1:**

**Input: s = "babad"**

**Output: "bab" Explanation: "aba" is also a valid answer.**

**Example 2:**

**Input: s = "cbbd"**

**Output: "bb"**

**Constraints: ● 1 <= s.length <= 1000 ● s consist of only digits and English letters.**

**Program:** class Solution:

```
  def longestPalindrome(self, s: str) -> str:
    def expandAroundCenter(left, right):
      while left >= 0 and right < len(s) and s[left] == s[right]:
        left -= 1
        right += 1
      return s[left + 1:right]

    if len(s) < 1:
      return ""

    longest = ""
    for i in range(len(s)):
      palindrome1 = expandAroundCenter(i, i)
      palindrome2 = expandAroundCenter(i, i + 1)
      longest = max(longest, palindrome1, palindrome2, key=len)

    return longest

# Test cases
solution = Solution()
print(solution.longestPalindrome("babad"))  # Output: "bab" or "aba"
print(solution.longestPalindrome("cbbd"))   # Output: "bb"
```

**Output:**

```
Output

bab
bb
```

**Timecomplexity: O(n)**