4.6 LONGEST PALINDROMIC SUBSTRING

Question:

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

AIM

To implement a Python program that finds the longest palindromic substring using dynamic programming.

ALGORITHM

1. Let n be the length of the string s.
2. Create a 2D boolean table dp[n][n] where dp[i][j] is True if the substring s[i:j+1] is a palindrome.
3. Initialize all substrings of length 1 as palindromes.
4. Check substrings of length 2 and mark them if both characters are equal.
5. For lengths ≥ 3, use the recurrence:
    - dp[i][j] = True if s[i] == s[j] and dp[i+1][j-1] == True
6. Track the start index and maximum length of the longest palindrome found.
7. Return the substring s[start:start+max_len].

PROGRAM

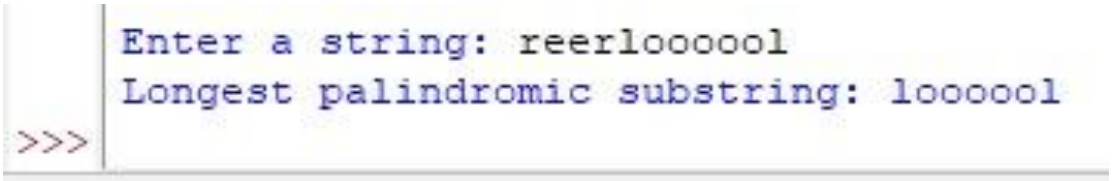
```
def longest_palindrome(s):
    start, max_len = 0, 0
    for i in range(len(s)):
        for j in range(i, len(s)):
            substr = s[i:j+1]
            if substr == substr[::-1] and len(substr) > max_len:
                start, max_len = i, len(substr)
    return s[start:start+max_len]

s = input("Enter a string: ")
print("Longest palindromic substring:", longest_palindrome(s))
```

Input:

Enter a string: reerloooool

Output:s



RESULT:

Thus the program is successfully executed, and the output is verified.

PERFORMANCE ANALYSIS:

- Time Complexity:
  - $O(n^2)$
- Space Complexity:
  - $O(n^2)$