Full Name:	Hideki Ikeda
Email:	hidekiai+hackerrank@gmail.com
Test Name:	**Mock Test**
Taken On:	12 Sep 2023 00:25:40 IST
Time Taken:	1 min 50 sec/ 22 min
Linkedin:	https://www.linkedin.com/in/hidekiai/
Invited by:	Ankush
Invited on:	12 Sep 2023 00:25:32 IST
Skills Score:	
Tags Score:	Algorithms 65/105 Core CS 65/105 Easy 65/105 Problem Solving 65/105 Strings 65/105 problem-solving 65/105

61.9%

65/105

scored in **Mock Test** in 1 min 50 sec on 12 Sep 2023 00:25:40 IST

Recruiter/Team Comments:

No Comments.

	Question Description	Time Taken	Score	Status
Q1	Palindrome Index > **Coding**	1 min 7 sec	65/ 105	✓

QUESTION 1

✓ Correct Answer

Score 65

Palindrome Index > Coding

Strings Algorithms Easy problem-solving Core CS Problem Solving

QUESTION DESCRIPTION

Given a string of lowercase letters in the range ascii[a-z], determine the index of a character that can be removed to make the string a palindrome. There may be more than one solution, but any will do. If the word is already a palindrome or there is no solution, return *-1*. Otherwise, return the index of a character to remove.

Example
$s = \text{"bcbc"}$

Either remove *'b'* at index 0 or *'c'* at index 3.

Function Description

Complete the *palindromeIndex* function in the editor below.

palindromeIndex has the following parameter(s):

- *string s:* a string to analyze

Returns

- *int:* the index of the character to remove or -1

Input Format

The first line contains an integer q, the number of queries.
Each of the next q lines contains a query string s.

Constraints

- $1 \le q \le 20$
- $1 \le \text{length of } s \le 10^5 + 5$
- All characters are in the range ascii[a-z].

Sample Input

```
STDIN   Function
-----   --------
3       q = 3
aaab    s = 'aaab' (first query)
baa     s = 'baa'  (second query)
aaa     s = 'aaa'  (third query)
```

Sample Output

```
3
0
-1
```

Explanation

Query 1: "aaab"
Removing *'b'* at index 3 results in a palindrome, so return 3.

Query 2: "baa"
Removing *'b'* at index 0 results in a palindrome, so return 0.

Query 3: "aaa"
This string is already a palindrome, so return -1. Removing any one of the characters would result in a palindrome, but this test comes first.

Note: The custom checker logic for this challenge is available here.

CANDIDATE ANSWER

Language used: **C++14**

```

/*
 * Complete the 'palindromeIndex' function below.
 *
 * The function is expected to return an INTEGER.
 * The function accepts STRING s as parameter.
 */
int palindromeIndex(string s) {
  // compare left and right string
  auto are_equal = [](const string &left, const string &right) {
    if (left.size() != right.size()) {
      return false;
    }
    // from left, we go from index=0, for right, we go from
index=right.size()-1
    // down to 0
    for (int i = 0; i < left.size(); ++i) {
      if (left[i] != right[right.size() - 1 - i]) {
        // immediately opt out soon as we find a mismatch
        return false;
      }
    }
    return true;
  };

  // in nature of palindrome, we have following characteristics:
  // * if the string is odd, the middle character is not important and we
only
  // compare left and right
  // * if the string is even, we compare left and right
  auto make_left_and_right = [](const string &s, string &left, string &right)
{
    if (s.size() % 2 == 0) {
      left = s.substr(0, s.size() / 2);
      right = s.substr(s.size() / 2, s.size() / 2);
    } else {
      left = s.substr(0, s.size() / 2);
      right = s.substr(s.size() / 2 + 1, s.size() / 2);
    }
  };

  // the edge case is when the string is aleady a palindrome:
 {
    string left, right;
    make_left_and_right(s, left, right);
    if (are_equal(left, right)) {
      // if already one, return -1
      return -1;
    }
  }

  // opmital of any single character will make the string a palindrome but
we'll
  // opt out on the first found.  We'll traverse from left to right and omit
  // one character, make left and right string, compare, and return the index
  for (int current_index = 0; current_index < s.size(); ++current_index) {
    string left, right;
    make_left_and_right(
        s.substr(0, current_index) + s.substr(current_index + 1), left,
right);
    if (are_equal(left, right)) {
      return current_index;
    }
  }
  return -1; // could not find any
}
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 1	Easy	Sample case	✓ Success	0	0.0587 sec	9 KB
Testcase 2	Medium	Hidden case	⊗ Wrong Answer	0	0.0381 sec	8.7 KB
Testcase 3	Medium	Hidden case	✓ Success	5	0.0274 sec	8.83 KB

Testcase 4	Medium	Hidden case	✓ Success	5	0.0275 sec	8.89 KB
Testcase 5	Medium	Hidden case	✓ Success	5	0.029 sec	8.68 KB
Testcase 6	Medium	Hidden case	✗ Terminated due to timeout	0	2.0025 sec	8.34 KB
Testcase 7	Medium	Hidden case	✓ Success	5	0.374 sec	9.28 KB
Testcase 8	Medium	Hidden case	✓ Success	5	1.8811 sec	9.02 KB
Testcase 9	Hard	Hidden case	✓ Success	10	0.766 sec	9.11 KB
Testcase 10	Hard	Hidden case	✓ Success	10	0.2112 sec	9.19 KB
Testcase 11	Hard	Hidden case	✗ Terminated due to timeout	0	2.0019 sec	8.77 KB
Testcase 12	Hard	Hidden case	✓ Success	10	0.038 sec	8.77 KB
Testcase 13	Hard	Hidden case	✗ Terminated due to timeout	0	2.0033 sec	8.63 KB
Testcase 14	Hard	Hidden case	✓ Success	10	1.2321 sec	8.71 KB
Testcase 15	Hard	Hidden case	✗ Terminated due to timeout	0	2.003 sec	8.88 KB

No Comments