You can view this report online at : https://www.hackerrank.com/x/tests/1108532/candidates/52124216/report

Full Name:	E O
Email:	okaforechezona1992@gmail.com
Test Name:	**Level 1 - Official Front-End Challenge**
Taken On:	3 May 2023 12:27:35 BST
Time Taken:	44 min 49 sec/ 45 min
Work Experience:	> 5 years
Invited by:	Landing.jobs
Invited on:	3 May 2023 12:27:04 BST
Skills Score:	HTML/CSS/JS 5/5 JavaScript (Basic) 50/50 Problem Solving (Basic) 8/50 REST API (Advanced) 3.33/5
Tags Score:	Algorithms 8/50 CSS 5/5 Closures 50/50 ES6 50/50 Easy 63/105 Front-End Development 5/5 HTML 5/5 HTML5 5/5 Interviewer Guidelines 8/50 JavaScript 50/50 Medium 3.33/5 Problem Solving 8/50 REST API 3.33/5 Strings 8/50 Web Development 5/5

60.3%

66/110

scored in **Level 1 - Official Front-End Challenge** in 44 min 49 sec on 3 May 2023 12:27:35 BST

Recruiter/Team Comments:

No Comments.

	Question Description	Time Taken	Score	Status
Q1	**JavaScript: Inventory List** > **Coding**	18 min 15 sec	50/ 50	✓
Q2	**Break a Palindrome** > **Coding**	24 min 29 sec	8/ 50	✓
Q3	**Start at Page Element** > **Multiple Choice**	37 sec	5/ 5	✓

QUESTION 1

Correct Answer

Score 50

JavaScript: Inventory List › Coding

ES6 | Easy | Closures | JavaScript

QUESTION DESCRIPTION

Implement a function, *inventoryList,* such that*:*

- it maintains the collection of all item names existing in an inventory, where each item is uniquely identified by a name.
- returns a new object, with three methods:
  - add(name) - The string *name* parameter is passed, and it is added to the collection. Items in the collection are distinct.
  - remove(name) - The string *name* parameter is passed, and this item is removed from the collection if it exists. If it does not exist, nothing happens.
  - getList() - This returns an array of names of items added so far. The names are returned in the order the corresponding items were added.

Your function implementation will be tested by a stubbed code on several input files. Each input file contains parameters for the function calls. The results of their execution will be printed to standard output by the provided code. The stubbed code joins the strings returned by the *getList* function by a comma and prints to the standard output. If *getList* returns an empty array, the stubbed code prints 'No Items'.

Constraints:

- The size of the collection will not exceed 10 at any point.
- All names passed to add(name) and remove(name) are non-empty.

▼ Input Format For Custom Testing

In the first line, there is an integer, *n*, denoting the number of operations to be performed.
Each line *i* of the *n* subsequent lines (where $0 \le i < n$) contains space-separated strings such that the first of them is a function name, and the remaining ones, if any, are parameters for that function.

▼ Sample Case 0

Sample Input For Custom Testing

```
5
add Shirt
add Trouser
getList
remove Shirt
getList
```

Sample Output

```
Shirt,Trouser
Trouser
```

Explanation

Items 'Shirt' and 'Trouser' are added by the *add* function. Then, *getList* is called, and the result is printed. Item 'Shirt' is removed by calling the *remove* function. Finally, *getList* is called, and the result is printed.

▼ Sample Case 1

Sample Input For Custom Testing

```
3
add Shirt
```

```
remove Trouser
getList
```

Sample Output

```
Shirt
```

Explanation

Item 'Shirt' is added by the *add* function. Then, *remove* is called with 'Trouser'. Since 'Trouser' does not exist, nothing happens. Finally, *getList* is called, and the result is printed.

INTERVIEWER GUIDELINES

tester's solution:

```
const assert = require('assert');
function inventoryList() {
  const names = [];
  return {
    add: (name) => {
      assert(name !== '');
      assert(!names.includes(name));
      assert(names.length <= 10);
      names.push(name);
    },
    remove: (name) => {
      const idx = names.indexOf(name);
      if (idx !== -1) {
        names.splice(idx, 1);
      }
    },
    getList: () => [...names],
  };
}
```

CANDIDATE ANSWER

Language used: **JavaScript (Node.js)**

```
1
2  let list = [];
3  function inventoryList() {
4    // write your code here
5    const getList = () => {
6      return list
7    }
8    const add =(name)=> {
9      list.push(name)
10   }
11   const remove = (name) => {
12       const index = list.indexOf(name);
13       if(index > -1) {
14         list.splice(index, 1)
15       }
16   }
17   return {getList, add, remove}
18 }
19
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED

Testcase 0	Easy	Sample case	Success	1	0.0941 sec	38.4 KB
Testcase 1	Easy	Sample case	Success	1	0.0517 sec	38.4 KB
Testcase 2	Easy	Sample case	Success	1	0.053 sec	37.9 KB
Testcase 3	Easy	Hidden case	Success	9	0.0549 sec	38.4 KB
Testcase 4	Medium	Hidden case	Success	11	0.057 sec	38.2 KB
Testcase 5	Easy	Hidden case	Success	9	0.0457 sec	38.6 KB
Testcase 6	Easy	Hidden case	Success	9	0.0547 sec	37.9 KB
Testcase 7	Easy	Hidden case	Success	9	0.0551 sec	38.2 KB

No Comments

QUESTION 2

Correct Answer

Score 8

Break a Palindrome › Coding

Strings | Algorithms | Easy | Problem Solving | Interviewer Guidelines

QUESTION DESCRIPTION

A palindrome reads the same from left or right, *mom* for example. There is a palindrome which must be modified, if possible. Change exactly one character of the string to another character in the range ascii[a-z] so that the string meets the following three conditions:

- The new string is lower alphabetically than the initial string.
- The new string is the lowest value string alphabetically that can be created from the original palindrome after making only one change.
- The new string is not a palindrome.

Return the new string, or, if it not possible to create a string meeting the criteria, return the string *IMPOSSIBLE*.

Example

palindromeStr = 'aaabbaaa'

- Possible strings lower alphabetically than *'aaabbaaa'* after one change are *['aaaabaaa', 'aaabaaaa']*.
- *'aaaabaaa'* is not a palindrome and is the lowest string that can be created from *palindromeStr*.

Function Description

Complete the function *breakPalindrome* in the editor below.

breakPalindrome has the following parameter(s):

string palindromeStr: the original string

Returns:

string: the resulting string, or *IMPOSSIBLE* if one cannot be formed

Constraints

- 1 ≤ length of *palindromeStr* ≤ 1000
- *palindromeStr* is a palindrome
- *palindromeStr* contains only lowercase English letters

▼ Input Format For Custom Testing

Locked stub code in the editor reads a single string, *palindromeStr,* from stdin and passes it to the function.

▼ Sample Case 0

Sample Input For Custom Testing

```
STDIN       Function
-----       --------
bab     →   palindromeStr  = 'bab'
```

Sample Output

```
aab
```

Explanation

- Possible strings lower alphabetically than *'bab'* after one change are *['aab', 'baa']*.
- *'aab'* is not a palindrome and is the lowest string that can be created from *palindromeStr*.

▼ Sample Case 1

Sample Input For Custom Testing

```
STDIN       Function
-----       --------
aaa     →   palindromeStr  = 'aaa'
```

Sample Output

```
IMPOSSIBLE
```

Explanation

- There are no strings lower alphabetically than *'aaa'* after one change.
- So, it is IMPOSSIBLE to create a string lower than *'aaa'*.

▼ Sample Case 2

Sample Input For Custom Testing

```
STDIN       Function
-----       --------
acca    →   palindromeStr  = 'acca'
```

Sample Output

```
aaca
```

Explanation

- Possible strings lower alphabetically than *'acca'* after one change are *['abca', 'aaca', 'acba', 'acaa']*.
- *'aaca'* is not a palindrome and is the lowest string that can be created from *palindromeStr*.

INTERVIEWER GUIDELINES

▼ Hint 1

Try to think if we want to make it lexicographically smallest, we must replace the first character, which we can replace with a smaller one.

▼ Hint 2

We should definitely replace a character with 'a' only, as it would make it minimal.

▼ Hint 3

Other than an all 'a' string, what are the possible cases when it is impossible to make a non-palindromic smaller string?

Ans - For odd lengths strings, if there are all 'a' except the center character as replacing the center character would still result in palindrome.

▼ Solution

Concepts Covered: Problem Solving, Greedy Algorithms

The problem tests the candidate's ability to use the basic knowledge of Greedy algorithms and solving it by greedily selecting the optimal strategy.

Optimal Solution:

We need to find the first character in the string which is not 'a' and replace it. We need it only for the first n/2 characters as it is a palindrome.

```
def breakPalindrome(palindromeStr):
    # store in a mutable structure
    sa = list(palindromeStr)
    # only consider 1st half (it's a palindrome initially)
    n = len(sa)//2 # returns integer floor of division
    # if any character is not 'a', replace it with 'a'
    # and return the new string
    for i in range(n):
        if not sa[i] == 'a':
            sa[i] = 'a'
            return ''.join(sa)
    # all characters are 'a' or the string is 1 character (always a
palindrome),
    # so it's not possible
    return 'IMPOSSIBLE'
```

Brute Force:

We can iterate over the string and try changing each character to any other character. For the new string, we can check the following two conditions:

1. The new string should not be a palindrome.

2. The new string should be alphabetically smaller than the original string.

We take the alphabetically smallest string as our answer among all such strings. If there is no string satisfying the above two conditions we return "IMPOSSIBLE" as the answer.

Time Complexity - O(N x N x 26)

▼ ,Complexity Analysis

Time Complexity - O(N) - We iterate over the array only once and therefore complexity is of the order O(N).

Space Complexity - O(1) - No extra space is required.

▼ Follow up Question

1. What would be the lexicographically smallest string which we can get if we have the optimal position to make changes in the given string?

Answer:

Since now we can rearrange characters as well, it would be a sorted version of a modified given string (consisting of at most one change at most one position).

2. Suppose you are allowed to rearrange the characters of the string now. What is the smallest possible string with at most one replacement that is smaller than the original string and is not a palindrome?

We can sort the characters of the string. A normal sort function would take O(N logN) time. We can do

counting sort instead, which runs in O(N) time and since there are only 26 characters in O(1) space. We can repeat the same above algorithm now. Note that here we need to check for non-'a' character till n and not just n/2.

Psuedo Code -

```
string breakPalindrome(string s) {
    int count[26];
    memset(count, 0, sizeof count);
    for(auto i:s)
        count[i - 'a']++;
    s.clear();
    for(int i=0; i<26; i++)
        while(count[i] > 0)
        {
            s.push_back(i + 'a');
            count[i]--;
        }
    int n = s.size();
    for(int i=0; i<n; i++)
        if(s[i]!='a')
        {
            s[i] = 'a';
            return s;
        }
    return "IMPOSSIBLE";
}
```

CANDIDATE ANSWER

Language used: **JavaScript (Node.js)**

```
/*
 * Complete the 'breakPalindrome' function below.
 *
 * The function is expected to return a STRING.
 * The function accepts STRING palindromeStr as parameter.
 */

function breakPalindrome(palindromeStr) {
    // Write your code here
    if(palindromeStr.length === 0){
        return "IMPOSSIBLE"
    }
    let sameChar = true;
    for(let i=palindromeStr.length-1; i >= 0; i--){
       if(palindromeStr !== palindromeStr[0]){
           sameChar = false;
           break;
       }
     }
     if(sameChar) {
         return "IMPOSSIBLE"
     }
     for(let i=palindromeStr.length-1; i >= 0; i--){
    //     let newWord = palindromeStr;
    //     if(i+1 > 0 && palindromeStr[i] !== palindromeStr[i+1] ) {
    //         newWord[i] = palindromeStr[i+1];
    //         if(newWord.split('').reverse().join('') !== newWord){
    //             return newWord
    //         }
    //     }
     }
```

```
32
33 }
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	Wrong Answer	0	0.0449 sec	38.4 KB
Testcase 1	Easy	Sample case	Wrong Answer	0	0.0564 sec	38.4 KB
Testcase 2	Easy	Sample case	Wrong Answer	0	0.0577 sec	38.2 KB
Testcase 3	Easy	Sample case	Success	4	0.0577 sec	38.5 KB
Testcase 4	Easy	Hidden case	Success	4	0.0436 sec	38.5 KB
Testcase 5	Easy	Hidden case	Wrong Answer	0	0.0483 sec	38.4 KB
Testcase 6	Easy	Hidden case	Wrong Answer	0	0.0515 sec	38.3 KB
Testcase 7	Easy	Hidden case	Wrong Answer	0	0.0572 sec	38.4 KB
Testcase 8	Hard	Sample case	Wrong Answer	0	0.0549 sec	38.2 KB
Testcase 9	Hard	Hidden case	Wrong Answer	0	0.05 sec	38.4 KB
Testcase 10	Hard	Hidden case	Wrong Answer	0	0.0487 sec	38.4 KB
Testcase 11	Hard	Hidden case	Wrong Answer	0	0.0511 sec	38.2 KB
Testcase 12	Hard	Hidden case	Wrong Answer	0	0.0493 sec	38.3 KB

No Comments

QUESTION 3

Correct Answer

Score 5

Start at Page Element › Multiple Choice

HTML5 | Front-End Development | Web Development | HTML | CSS | Easy

QUESTION DESCRIPTION

When a web page is loaded, the user's cursor should be in the only text field on the page. How is this accomplished using HTML5?

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

- <input type="text" focus=True>
- ✓ <input type="text" autofocus>
- <input type="text" focus=onPageLoad>
- It cannot be accomplished with only HTML5.

No Comments

QUESTION 4

Correct Answer

Score 3.33

REST Services › Multiple Choice

REST API | Medium

QUESTION DESCRIPTION

Which of the following statements are true about REST?

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

- [x] Logical URLs should be used instead of physical URLs
- [x] Actual URLs must always be used in a REST response.
- [ ] A paging technique should be used if the output data is small.
- [x] GET requests must be read-only.
- [ ] The output format can be changed.
- [ ] POST requests must be read-only.

No Comments