Accenture Sections	Information	Questions and Time
Cognitive Ability	• English Ability • Critical Thinking and Problem Solving • Abstract Reasoning	50 Ques in 50 mins
Technical Assessment	• Common Application and MS Office • Pseudo Code • Fundamental of Networking, Security and Cloud	40 Ques in 40 mins
Coding Round	• C • C++ • Dot Net • JAVA • Python	2 Ques in 45 mins

DEBUG WITH SHUBHAM

Accenture Technical Assessment Detailed Overview

17-SEP-2024 Coding Question


https://www.youtube.com/@DebugWithShubham

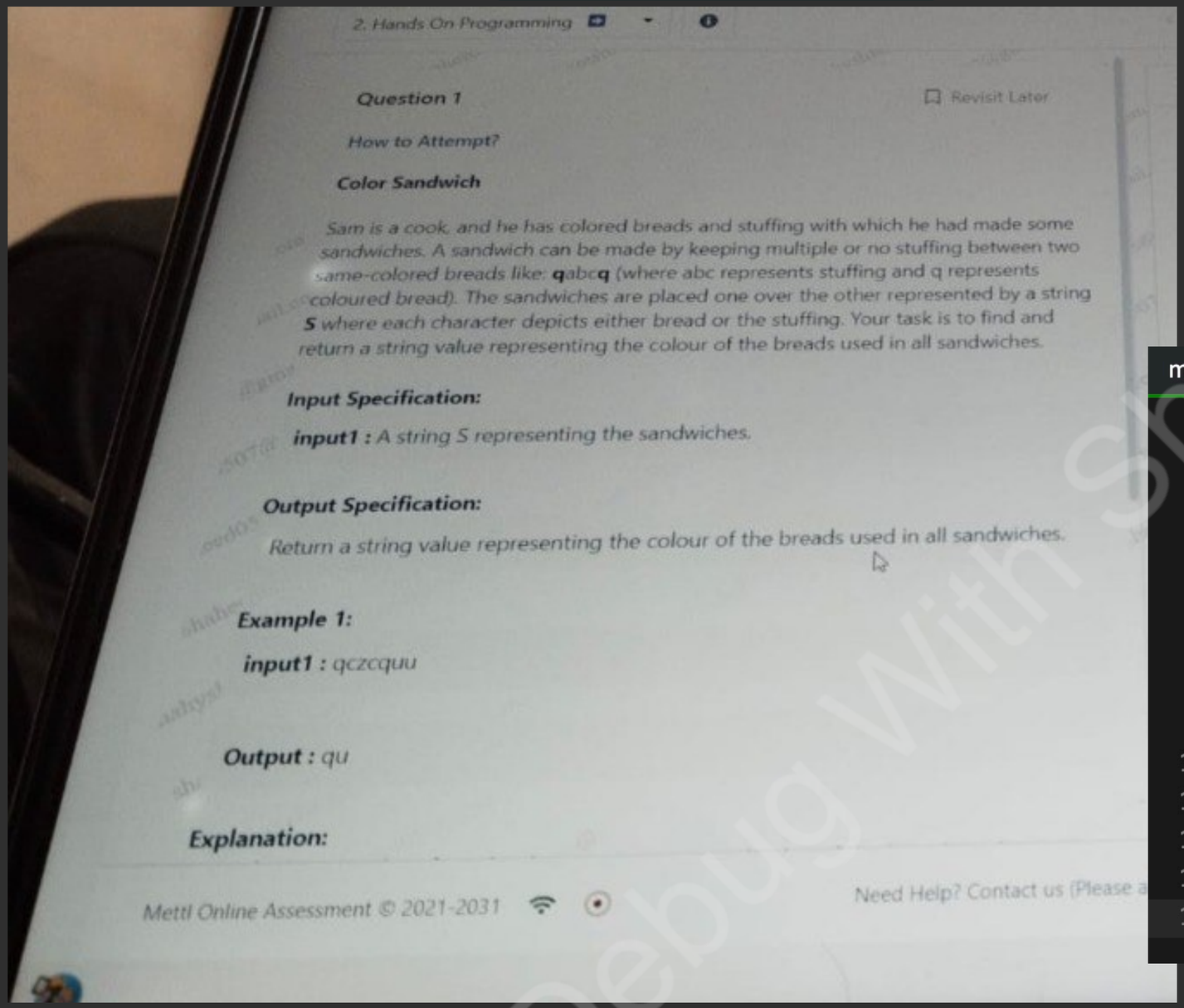

https://www.linkedin.com/in/debugwithshubham/


https://www.instagram.com/debugwithshubham/


https://topmate.io/debugwithshubham


https://t.me/debugwithshubham

Question-1



2. Hands On Programming

Question 1

Revisit Later

How to Attempt?

Color Sandwich

Sam is a cook, and he has colored breads and stuffing with which he had made some sandwiches. A sandwich can be made by keeping multiple or no stuffing between two same-colored breads like: **q**abc**q** (where abc represents stuffing and q represents coloured bread). The sandwiches are placed one over the other represented by a string **S** where each character depicts either bread or the stuffing. Your task is to find and return a string value representing the colour of the breads used in all sandwiches.

Input Specification:

input1 : A string S representing the sandwiches.

Output Specification:

Return a string value representing the colour of the breads used in all sandwiches.

Example 1:

input1 : qczcquu

Output : qu

Explanation:

Mettl Online Assessment © 2021-2031

Need Help? Contact us (Please a

Brute- Python

main.py +

```python
def find_bread_colors(s):
    bread_colors = set()
    n = len(s)
    for i in range(n):
        start_bread = s[i]
        for j in range(i + 1, n):
            if s[j] == start_bread:
                bread_colors.add(start_bread)
                break
    return ''.join(sorted(bread_colors))
input_str = "qezcquu"
output = find_bread_colors(input_str)
print(output)

```

Brute- JAVA

```java
import java.util.HashSet;
import java.util.Set;
import java.util.TreeSet;
public class SandwichColors {
    public static String findBreadColors(String s) {
        Set<Character> breadColors = new TreeSet<>();
        int n = s.length();
        for (int i = 0; i < n; i++) {
            char startBread = s.charAt(i);
            for (int j = i + 1; j < n; j++) {
                if (s.charAt(j) == startBread) {
                    breadColors.add(startBread);
                    break;
                }
            }
        }
        StringBuilder result = new StringBuilder();
        for (char c : breadColors) {
            result.append(c);
        }
        return result.toString();
    }
    public static void main(String[] args) {
        String inputStr = "qezcquu";
        String output = findBreadColors(inputStr);
        System.out.println(output);
    }
}
```

Brute- C++

main.cpp

```cpp
#include <iostream>
#include <set>
#include <string>
std::string findBreadColors(const std::string& s) {
    std::set<char> breadColors;
    int n = s.length();
    for (int i = 0; i < n; ++i) {
        char startBread = s[i];
        for (int j = i + 1; j < n; ++j) {
            if (s[j] == startBread) {
                breadColors.insert(startBread);
                break;
            }
        }
    }
    std::string result;
    for (char c : breadColors) {
        result += c;
    }
    return result;
}
int main() {
    std::string inputStr = "qezcquu";
    std::string output = findBreadColors(inputStr);
    std::cout << output << std::endl;
    return 0;
}
```

Best- Python

main.py

```python
def find_bread_colors(s):
    bread_colors = set()
    n = len(s)
    i = 0
    while i < n:
        current_bread = s[i]
        j = i + 1
        while j < n and s[j] != current_bread:
            j += 1
        if j < n:
            bread_colors.add(current_bread)
        i = j + 1
    return ''.join(sorted(bread_colors))
print(find_bread_colors("qezcquu"))

```

Best- Java

Main.java

```java
import java.util.Set;
import java.util.TreeSet;

public class SandwichColors {
    public static String findBreadColors(String s) {
        Set<Character> breadColors = new TreeSet<>();
        int n = s.length();
        int i = 0;
        while (i < n) {
            char currentBread = s.charAt(i);
            int j = i + 1;
            while (j < n && s.charAt(j) != currentBread) {
                j++;
            }
            if (j < n) {
                breadColors.add(currentBread);
            }
            i = j + 1;
        }
        StringBuilder result = new StringBuilder();
        for (char c : breadColors) {
            result.append(c);
        }
        return result.toString();
    }

    public static void main(String[] args) {
        String inputStr = "qezcquu";
        String output = findBreadColors(inputStr);
        System.out.println(output);
    }
}

```

Best-C++

main.cpp

```cpp
#include <iostream>
#include <set>
#include <string>
std::string findBreadColors(const std::string& s) {
    std::set<char> breadColors;
    int n = s.length();
    int i = 0;
    while (i < n) {
        char currentBread = s[i];
        int j = i + 1;
        while (j < n && s[j] != currentBread) {
            ++j;
        }
        if (j < n) {
            breadColors.insert(currentBread);
        }
        i = j + 1;
    }
    std::string result;
    for (char c : breadColors) {
        result += c;
    }
    return result;
}

int main() {
    std::string inputStr = "qezcquu";
    std::string output = findBreadColors(inputStr);
    std::cout << output << std::endl;
    return 0;
}

```

Question-2

2. Hands On Programming

Question 2

Revisit Later

How to Attempt?

String Program

Alice is working on a system that processes text inputs from users. One of the requirements is that the text must not contain the substring "ppp" - three consecutive 'p' characters. If this substring appears, the text is rejected.

To make the input text acceptable, Alice needs to remove characters in such a way that the substring "ppp" does not exist.

Your task is to help Alice find and return an integer value representing the minimum number of characters that need to be removed from the text to meet this requirement.

Note:

- *The string only contains lowercase alphabetic characters.*
- *You can delete characters in random positions (not necessarily consecutive).*

Input Specification:

input1 : A string S, representing the text inputs from users.

Output Specification:

Return an integer value representing the minimum number of characters that need

Python

main.py +

```python
def min_removals_to_avoid_ppp(S):
    n = len(S)
    removals = 0
    i = 0
    while i < n:
        if i + 2 < n and S[i] == 'p' and S[i+1] == 'p' and S[i+2] == 'p':
            removals += 1
            i += 1
        else:
            i += 1
    return removals
S = "pppabppp"
print(min_removals_to_avoid_ppp(S))

```

JAVA

Main.java

Share Run

```java
public class MinRemovalsToAvoidPPP {
    public static int minRemovalsToAvoidPPP(String S) {
        int n = S.length();
        int removals = 0;
        int i = 0;
        while (i < n) {
            if (i + 2 < n && S.charAt(i) == 'p' && S.charAt(i + 1) == 'p'
                && S.charAt(i + 2) == 'p') {
                removals++;
                i += 1;
            } else {
                i += 1;
            }
        }
        return removals;
    }
    public static void main(String[] args) {
        String S = "pppabppp";
        int result = minRemovalsToAvoidPPP(S);
        System.out.println(result);
    }
}
```

C++

main.cpp

Share Run

```cpp
#include <iostream>
#include <string>
int minRemovalsToAvoidPPP(const std::string& S) {
    int n = S.length();
    int removals = 0;
    int i = 0;
    while (i < n) {
        if (i + 2 < n && S[i] == 'p' && S[i + 1] == 'p' && S[i + 2] == 'p')
            {
            removals++;
            i += 1;
        } else {
            i += 1;
        }
    }
    return removals;
}
int main() {
    std::string S = "pppabppp";
    int result = minRemovalsToAvoidPPP(S);
    std::cout << result << std::endl;
    return 0;
}
```