Accenture Sections	Information	Questions and Time
Cognitive Ability	English AbilityCritical Thinking and Problem SolvingAbstract 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/

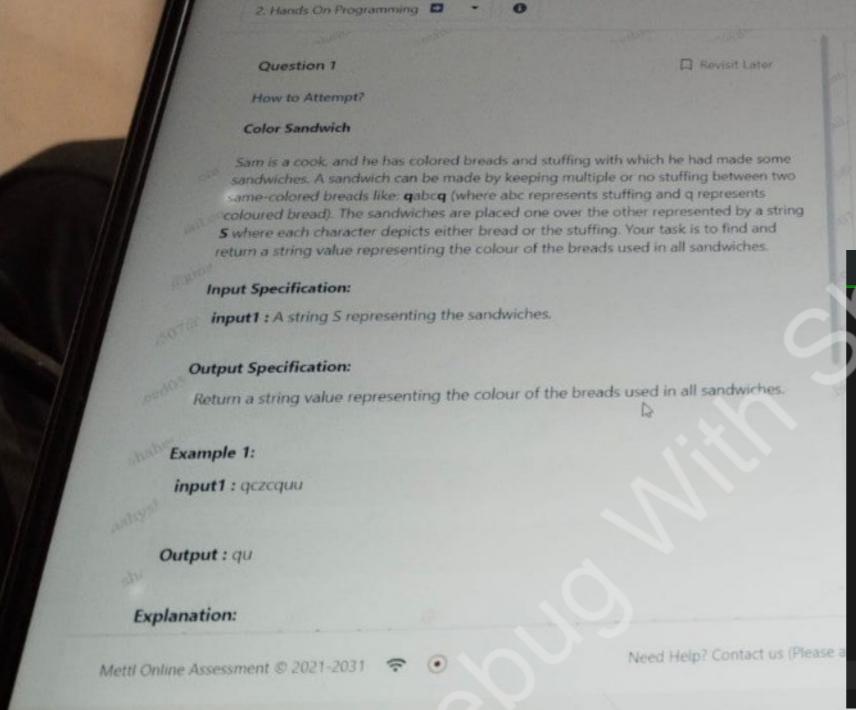


https://topmate.io/debugwithshubham



https://t.me/debugwithshubham

Question-1



Brute- Python

```
main.py
    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)
 8
                     break
        return ''.join(sorted(bread_colors))
10
    input_str = "qezcquu"
    output = find_bread_colors(input_str)
    print(output)
14
```

Brute- 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
 1 #include <iostream>
 2 #include <set>
 3 #include <string</pre>
 4 std::string findBreadColors(const std::string& s) {
        std::set<char> breadColors;
 5
        int n = s.length();
 6
        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;
15
        std::string result;
16
        for (char c : breadColors) {
17 -
            result += c;
18
19
20
        return result;
21 }
22 int main() {
        std::string inputStr = "qezcquu";
23
        std::string output = findBreadColors(inputStr);
24
        std::cout << output << std::endl;</pre>
25
26
        return 0;
27 }
28
```

Best-Python

```
main.py
 1 - def find_bread_colors(s):
         bread_colors = set()
         n = len(s)
         i = 0
        while i < n:
 5 -
             current_bread = s[i]
             j = i + 1
             while j < n and s[j] != current_bread:</pre>
 8 ~
                 j += 1
 9
             if j < n:
10 -
                 bread_colors.add(current_bread)
11
             i = j + 1
12
         return ''.join(sorted(bread_colors))
13
    print(find_bread_colors("qezcquu"))
15
```

Best-Java

Main.java





```
1 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;
 8
            while (i < n) {
 9
                char currentBread = s.charAt(i);
10
                int j = i + 1;
                while (j < n && s.charAt(j) != currentBread) {</pre>
                     j++;
                if (j < n) {
                    breadColors.add(currentBread);
17
                i = j + 1;
18
19
            StringBuilder result = new StringBuilder();
20
            for (char c : breadColors) {
21 -
                result.append(c);
22
23
            return result.toString();
24
25
        }
26
        public static void main(String[] args) {
27
           String inputStr = "qezcquu";
28
           String output = findBreadColors(inputStr);
29
           System.out.println(output);
30
31
32 }
33
```

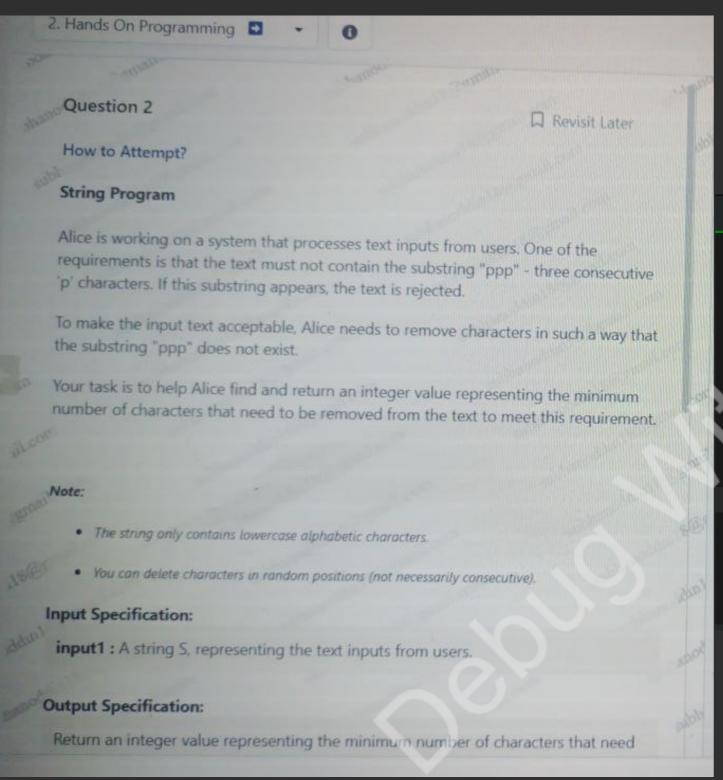
Best-C++





```
1 #include <iostream>
 2 #include <set>
 3 #include <string>
 4 std::string findBreadColors(const std::string& s) {
        std::set<char> breadColors;
 5
        int n = s.length();
 6
        int i = 0;
 7
        while (i < n) {
 8
            char currentBread = s[i];
 9
            int j = i + 1;
10
            while (j < n && s[j] != currentBread) {</pre>
11 -
12
                 ++j;
13
            if (j < n)
14 -
                breadColors.insert(currentBread);
15
16
17
18
        std::string result;
19
        for (char c : breadColors) {
20
21
            result += c;
        return result;
    int main() {
27
        std::string inputStr = "qezcquu";
        std::string output = findBreadColors(inputStr);
28
29
        std::cout << output << std::endl;</pre>
30
        return 0;
31 }
32
```

Question-2



Python



C++

```
∝ Share
Main.java
                                                                         Run
 1 - public class MinRemovalsToAvoidPPP {
        public static int minRemovalsToAvoidPPP(String S) {
 2
 3
            int n = S.length();
 4
            int removals = 0;
            int i = 0;
 5
 6
            while (i < n) {
 7 -
                if (i + 2 < n && S.charAt(i) == 'p' && S.charAt(i + 1) == 'p'</pre>
                    && S.charAt(i + 2) == 'p') {
 8
                    removals++;
 9
                    i += 1;
                } else {
10
                    i += 1;
11
12
13
14
            return removals;
15
       public static void main(String[] args) {
16
            String S = "pppabppp";
17
            int result = minRemovalsToAvoidPPP(S);
18
            System.out.println(result);
19
20
        }
21 }
22
```

```
-;ċ;-
                                                           ∝ Share
main.cpp
                                                                         Run
 1 #include <iostream>
 2 #include <string>
 3 int minRemovalsToAvoidPPP(const std::string& S) {
        int n = S.length();
 4
        int removals = 0;
 5
 6
        int i = 0;
        while (i < n) {
 7 -
             if (i + 2 < n && S[i] == 'p' && S[i + 1] == 'p' && S[i + 2] == 'p')
 8
                removals++;
                i += 1;
            } else {
                i += 1;
            }
        return removals;
16 }
17 int main() {
        std::string S = "pppabppp";
18
        int result = minRemovalsToAvoidPPP(S);
19
        std::cout << result << std::endl;</pre>
20
        return 0;
21
22 }
23
```