

DEBUG WITH SHUBHAM

Accenture Technical interviews Detailed Overview

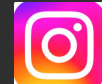
15-Oct-2024 Coding Interview Questions



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



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



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



<https://topmate.io/debugwithshubham>



<https://t.me/debugwithshubham>

ALL SOLUTION (C++, JAVA, PYTHON) UPLOADED IN GITHUB WITH QUESTION NAME

Question 1

🔖 Revisit Later

How to Attempt?

Rebound Height

Daniel has a ball. He wants to find the ball's rebound height, which he dropped from height H with an initial velocity V . After the N^{th} rebound the final velocity of the ball is V_n . Your task is to help him find and return an integer value representing the height to which the ball rebounds after N bounces.

Note:

- $H' = H \times e^{2n}$, where H' is the rebound height, H is the initial height, e is the coefficient of restitution and n is the number of bounces.
- $e^n = V/V_n$, where V is the initial velocity and V_n is the final velocity

Input Specification:

main.py +

```
1
2 def rebound_height(H, V, Vn,N):
3     e = V / Vn
4     en = e ** (2*N)
5     rebound_height = H * en
6     return int(rebound_height)
7 H = 10
8 V = 15
9 Vn = 5
10 N = 3
11 print(rebound_height(H, V, Vn,N))
12 |
```

Main.java



Share

Run

```
1 public class ReboundHeightCalculator {
2     public static int reboundHeight(double H, double V, double Vn,
3         int N) {
4         double e = V / Vn;
5         double en = Math.pow(e, 2 * N);
6         double reboundHeight = H * en;
7         return (int) reboundHeight;
8     }
9     public static void main(String[] args) {
10         double H = 10;
11         double V = 15;
12         double Vn = 5;
13         int N = 3;
14         System.out.println(reboundHeight(H, V, Vn, N));
15     }
16 |
```

main.cpp



Share

```
1 #include <iostream>
2 #include <cmath>
3 using namespace std;
4 int reboundHeight(double H, double V, double Vn, int N) {
5     double e = V / Vn;
6     double en = pow(e, 2 * N);
7     double reboundHeight = H * en;
8     return (int) reboundHeight;
9 }
10 int main() {
11     double H = 10;
12     double V = 15;
13     double Vn = 5;
14     int N = 3;
15     cout << reboundHeight(H, V, Vn, N) << endl;
16     return 0;
17 }
18 |
```

File display

1. Coding

Question 2

How to Attempt?

Alice and String

Alice has a peculiar fondness for strings without any interruption considers "." as an interruption. You are given a string **S** and you to find and return the length of the longest uninterrupted substr match alices' fondness.

Input Specification:

input1 : A string value S.

Output Specification:

Return an integer value representing the length of the longest uninterrupted substring to match alices' fondness.

Example 1:

With Split

```
main.py +
1 def longest_uninterrupted_substring(S):
2     substrings = S.split('.')
3     max_length = 0
4     for substring in substrings:
5         if len(substring) > max_length:
6             max_length = len(substring)
7     return max_length
8
9 S = "this.is.a.debugwithshubham"
10 output = longest_uninterrupted_substring(S)
11 print(output)
12
```

```
Main.java
1 public class LongestUninterruptedSubstring {
2     public static int longestUninterruptedSubstring(String S) {
3
4         String[] substrings = S.split("\\.");
5         int maxLength = 0;
6         for (String substring : substrings) {
7             if (substring.length() > maxLength) {
8                 maxLength = substring.length();
9             }
10        }
11        return maxLength;
12    }
13
14    public static void main(String[] args) {
15        String S = "this.is.a.debugwithshubham";
16        int output = longestUninterruptedSubstring(S);
17        System.out.println(output);
18    }
19 }
```

```
main.cpp
1 #include <iostream>
2 #include <string>
3 #include <vector>
4 #include <sstream>
5 using namespace std;
6 vector<string> split(const string &S, char delimiter) {
7     vector<string> substrings;
8     stringstream ss(S);
9     string temp;
10    while (getline(ss, temp, delimiter)) {
11        substrings.push_back(temp);
12    }
13    return substrings;
14 }
15 int longestUninterruptedSubstring(const string &S) {
16     vector<string> substrings = split(S, '.');
17     int maxLength = 0;
18     for (const string &substring : substrings) {
19         if (substring.length() > maxLength) {
20             maxLength = substring.length();
21         }
22     }
23     return maxLength;
24 }
25 int main() {
26     string S = "this.is.a.debugwithshubham";
27     int output = longestUninterruptedSubstring(S);
28     cout << output << endl;
29     return 0;
30 }
31
```

```
main.py +
1 def longest_uninterrupted_substring(S):
2     max_length = 0
3     current_length = 0
4     for char in S:
5         if char == '.':
6             max_length = max(max_length, current_length)
7             current_length = 0
8         else:
9             current_length += 1
10    max_length = max(max_length, current_length)
11    return max_length
12 S = "this.is.a.debugwithshubham"
13 output = longest_uninterrupted_substring(S)
14 print(output)
```

Main.java

```
1 public class LongestUninterruptedSubstring {
2     public static int longestUninterruptedSubstring(String S) {
3         int maxLength = 0;
4         int currentLength = 0;
5         for (int i = 0; i < S.length(); i++) {
6             if (S.charAt(i) == '.') {
7                 maxLength = Math.max(maxLength, currentLength);
8                 currentLength = 0;
9             } else {
10                currentLength++;
11            }
12        }
13        maxLength = Math.max(maxLength, currentLength);
14
15        return maxLength;
16    }
17    public static void main(String[] args) {
18        String S = "this.is.a.debugwithshubham";
19        int output = longestUninterruptedSubstring(S);
20        System.out.println(output);
21    }
22 }
23 |
```

main.cpp

```
1 #include <iostream>
2 #include <string>
3 using namespace std;
4 int longestUninterruptedSubstring(const string &S) {
5     int maxLength = 0;
6     int currentLength = 0;
7     for (char c : S) {
8         if (c == '.') {
9             maxLength = max(maxLength, currentLength);
10            currentLength = 0;
11        } else {
12            currentLength++;
13        }
14    }
15    maxLength = max(maxLength, currentLength);
16    return maxLength;
17 }
18 int main() {
19     string S = "this.is.a.debugwithshubham";
20     int output = longestUninterruptedSubstring(S);
21     cout << output << endl;
22     return 0;
23 }
24
```