

ROUND - 1 QUESTIONS AND ANSWERS

Question - 1 :

The screenshot shows a web browser with multiple tabs. The active tab is 'check.pcc.events/round1'. The page displays a coding problem on '1Easy' with a timer showing 00 Days, 01 Hours, 26 Minutes, and 18 Seconds. The problem text is: 'Given an expression string x. Examine whether the pairs and the orders of "(", ")", "{", "}", "[", "]", "]" are correct in exp. For example, the function should return 'true' for exp = "{()}{}{({})}()" and 'false' for exp = "{()}".' The input guide says: 'Insert you answer(which will be string) without quotes like :{ ([]) } .'. Example 1 shows Input: {{[()]}, Output: true, and Explanation: {{[()]}. Same kind of brackets can form balanced pairs, with 0 number of unbalanced bracket. To the right, the C++ solution is shown, using a stack to check for balanced brackets. A red arrow points to the browser's address bar.

1Easy

Given an expression string x. Examine whether the pairs and the orders of "(", ")", "{", "}", "[", "]", "]" are correct in exp. For example, the function should return 'true' for exp = "{()}{}{({})}()" and 'false' for exp = "{()}".

Input Guide :
Insert you answer(which will be string) without quotes like :{ ([]) } .

Example 1:
Input : {{[()]}
Output : true
Explanation : {{[()]}. Same kind of brackets can form balanced pairs, with 0 number of unbalanced bracket.

```
1 #include<iostream>
2 #include<vector>
3 #include<string>
4 #include<stack>
5 using namespace std;
6
7
8
9 class Solution
10 {
11 public:
12     bool ispar(string x)
13     {
14         stack<int> s;
15         int i=0;
16         while(i<x.size()){
17             if(x[i]=='('||x[i]=='{'||x[i]=='['){
18                 if(x[i]=='('){
19                     s.push(-1);
20                 }
21                 if(x[i]=='['){
22                     s.push(0);
23                 }
24                 if(x[i]=='{'){
25                     s.push(1);
26                 }
27             }
28             else{
29                 if(s.size()==0){
30                     return false;
31                 }
32                 else{
33                     if(x[i]==')'&&s.top()!=-1){
34                         return false;
35                     }
36                     if(x[i]=='}'&&s.top()!=0){
37                         return false;
38                     }
39                     if(x[i]==']'&&s.top()!=1){
40                         return false;
41                     }
42                 }
43             }
44             i++;
45         }
46         return s.empty();
47     }
48 }
```

Answer: {{[({})]}}

Question - 2 :

Medium

Given two strings *s* and *t*, return **true** if they are equal when both are typed into empty text editors. '#' means a backspace character.

Note that after backspacing an empty text, the text will continue empty.

Input Guide :

Insert your answer (which will be 2 strings) without quotes.

Insert first string without quotes then in a different line

Insert the second string without quotes.

asdf#f

asd#df

NOTE : Do not forget to change the line for second string input.

```
1 #include<iostream>
2 #include<vector>
3 #include<cstring>
4 #include<stack>
5 #include<algorithm>
6 #include<unordered_map>
7 using namespace std;
8
9 class Solution {
10 public:
11     bool backspaceCompare(string s, string t) {
12         stack<char> st;
13         int n=s.size();
14         for(int i=0;i<n;i++){
15             if(st.empty()){
16                 st.push(s[i]);
17                 continue;
18             }
19             if(s[i]=='#'){
20                 st.pop();
21                 continue;
22             }
23             st.push(s[i]);
24         }
25         string ans1="";
26         while(!st.empty()){
27             ans1.push_back(st.top());
28             st.pop();
29         }
30         int m=t.size();
31         for(int i=0;i<m;i++){
32             if(st.empty()){
33                 st.push(t[i]);
34                 continue;
35             }
36             if(t[i]=='#'){
37                 st.pop();
38                 continue;
39             }
40             st.push(t[i]);
41         }
```

Answer: : S =“##abc#” I = “ab”

Question - 3 :

The screenshot shows a web browser window with a coding problem titled "Restore IP Addresses". The problem is marked as "Hard". The description states: "A valid IP address consists of exactly four integers separated by single dots. Each integer is between 0 and 255 (inclusive) and cannot have leading zeros. For example, '0.1.2.201' and '192.168.1.1' are valid IP addresses, but '0.011.255.245', '192.168.1.312' and '192.168@1.1' are invalid IP addresses. Given a string s containing only digits, return all possible valid IP addresses that can be formed by inserting dots into s. You are not allowed to reorder or remove any digits in s. You may return the valid IP addresses in any order." The input guide says: "Insert you ans (which will be numeric string) without quotes like : 2344567 .". Example 1: "Input: s = '25525511132', Output: ['255.255.11.132', '255.255.111.32']". On the right, a C++ code solution is shown, using a recursive helper function to generate valid IP addresses. The left sidebar shows a timer: 00 Days, 01 Hours, 29 Minutes, and 41 Seconds.

4Hard

A valid IP address consists of exactly four integers separated by single dots. Each integer is between 0 and 255 (inclusive) and cannot have leading zeros.

- For example, "0.1.2.201" and "192.168.1.1" are valid IP addresses, but "0.011.255.245", "192.168.1.312" and "192.168@1.1" are invalid IP addresses.

Given a string `s` containing only digits, return *all possible valid IP addresses that can be formed by inserting dots into s*. You are **not** allowed to reorder or remove any digits in `s`. You may return the valid IP addresses in **any** order.

Input Guide :
Insert you ans (which will be numeric string) without quotes like : 2344567 .

Example 1:
Input: `s = "25525511132"`
Output: `["255.255.11.132", "255.255.111.32"]`

```
1 #include<iostream>
2 #include<vector>
3 #include<string>
4 #include<stack>
5 using namespace std;
6
7 class Solution {
8 public:
9     vector<string> ans;
10    bool valid(string s){
11        if(s[0]=='0'){
12            return false;
13        }
14        int v=stoi(s);
15        if(v>255){
16            return false;
17        }
18        return true;
19    }
20    void helper(string &s,int i,int part,string res){
21        if(s.size()==i||part==4){
22            if(s.size()==i&&part==4){
23                ans.push_back(res.substr(0,res.size()-1));
24                return;
25            }
26            return;
27        }
28        if(s.size()-i>=1&&valid(s.substr(i,1)){
29            helper(s,i+1,part+1,res+s[i]+".");
30        }
31        if(s.size()-i>=2&&valid(s.substr(i,2)){
32            helper(s,i+2,part+1,res+s.substr(i,2)+".");
33        }
34        if(s.size()-i>=3&&valid(s.substr(i,3)){
35            helper(s,i+3,part+1,res+s.substr(i,3)+".");
36        }
37        return;
38    }
39    vector<string> restoreIpAddresses(string s) {
40        helper(s,0,0,"");
41        return ans;
42    }
43 }
```

Answer : 0000

Question - 4 :

The screenshot shows a web browser window with a coding problem on the left and its C++ solution on the right. The problem is titled "3Easy" and asks to delete characters appearing more than once consecutively from a string S. The input guide shows an example with S = "aabb" and output "ab". The explanation states that 'a' at the 2nd position is appearing 2nd time consecutively, and 'b' at the 4th position is appearing 2nd time consecutively. The solution code is a C++ program that uses a recursive helper function to remove consecutive duplicate characters from a string.

3Easy

Given a string **S** delete the characters which are appearing more than once consecutively.

Input Guide :
Insert you ans (which will be string) without quotes like : aacbdd .

Example 1:
Input: S = aabb
Output: ab
Explanation:
'a' at 2nd position is appearing 2nd time consecutively. Similar explanation for b at 4th position.

Example 2:

```
1 #include<iostream>
2 #include<vector>
3 #include<string>
4 #include<stack>
5 using namespace std;
6
7
8 class Solution{
9 public:
10 void helper(string& S, int index, string& ans){
11     if(index==S.size()-1){
12         if(S[index]!=S[index-1]){
13             ans.push_back(S[index]);
14         }
15         return;
16     }
17     if(index==S.size()-1){
18         ans.push_back(S[index]);
19         return;
20     }
21     if(S[index]==S[index+1]){
22         helper(S, index+1, ans);
23     }
24     else{
25         ans.push_back(S[index]);
26         helper(S, index+1, ans);
27         return ;
28     }
29 }
30 string removeConsecutiveCharacter(string S)
31 {
32     string ans;
33     helper(S, 0, ans);
34     return ans;
35 };
36 };
37
38 int main()
39 {
40 }
```

Answer : adadaaddddad

Question - 5 :

9 Medium

Given string `num` representing a non-negative integer `num`, and an integer `k`, return the *smallest possible integer after removing `k` digits from `num`*.

Input Guide :
Insert Numeric string in the first line like **2345466** and in next line insert non negative integer `k`

NOTE : Do not forget to change the line

Example 1:
Input: `num = "1432219"`, `k = 3`
Output: `"1219"`
Explanation: Remove the three digits 4, 3, and 2 to form the new number 1219 which is the smallest.

```
1 #include<iostream>
2 #include<vector>
3 #include<string>
4 #include<stack>
5 #include<algorithm>
6 using namespace std;
7
8 class Solution {
9 public:
10     string removeKdigits(string num, int k){
11         int n=num.size();
12         stack<int>s;
13         if(num.size()==k){
14             return "0";
15         }
16         for(int i=0;i<n;i++){
17             if(s.empty()){
18                 s.push(num[i]);
19                 continue;
20             }
21             else{
22                 if(s.top()>num[i]){
23                     while(!s.empty()&&s.top()>num[i]&&k!=0){
24                         s.pop();
25                         k--;
26                     }
27                 }
28                 s.push(num[i]);
29             }
30         }
31         string ans="";
32         while(!s.empty()){
33             ans.push_back(s.top());
34             s.pop();
35         }
36         while(ans.size()!=0&&ans.back()=='0'){
37             ans.pop_back();
38         }
39         reverse(ans.begin(),ans.end());
40     }
```

Answer : "123"
k=1

Question - 6 :

The screenshot shows a web browser with multiple tabs. The active tab is 'check.pcc.events/next'. The page displays a coding problem titled '2Hard'. The problem description is: 'Given a string S consisting of only opening and closing curly brackets '{' and '}', find out the minimum number of reversals required to convert the string into a balanced expression. A reversal means changing '{' to '}' or vice-versa.' The input guide states: 'Insert you ans (which will be string) without quotes like: {{{}}}'. Example 1: Input: S = {{{}}}{ Output: 2. Explanation: One way to balance is: {{{}}}{. There is no balanced sequence that can be formed in lesser reversals. Example 2: Input: S = {{{}}}{ Output: -1. To the right of the problem description is a code editor showing a C++ solution. The code defines a function 'countRev' that uses a stack to count the minimum number of reversals required to balance a string of curly brackets. The code is as follows:

```
1 #include<iostream>
2 #include<vector>
3 #include<string>
4 #include<stack>
5 using namespace std;
6
7 int countRev (string s);
8
9 int main()
10 {
11     string s;
12     cin >> s;
13     cout << countRev (s) << '\n';
14 }
15
16 int countRev (string s)
17 {
18     stack<char> st;
19     if(s.size()%2!=0){
20         return -1;
21     }
22     for(int i=0;i<s.size();i++){
23         if(st.empty()){
24             st.push(s[i]);
25         }
26         else{
27             if(st.top()!='{'&&s[i]=='}'){
28                 st.pop();
29             }else{
30                 st.push(s[i]);
31             }
32         }
33     }
34     return st.size()/2;
35 }
36 }
```

Answer : :"}{{{}}}{{"

Question - 7 :

The screenshot shows a web browser with multiple tabs. The active tab is 'check.pcc.events/next'. The page displays a coding problem titled '6/ Medium'. The problem asks to find the minimum depth of a binary tree. The input is a string of characters representing nodes and null nodes. The output is the minimum depth. The solution is provided in C++ code.

6/ Medium

Given a binary tree, find its minimum depth.

The minimum depth is the number of nodes along the shortest path from the root node down to the nearest leaf node.

Note: A leaf is a node with no children.

Input Guide :

Insert you answer (which will be tree) you have to insert tree in inline where nodes are separated by white spaces and non existing nodes(null node) will be inserted as N like : **1 2 3 N 4 5 6** (recursive input) .

Example 1:

Input: root = 3 9 20 N N 15 7
Output: 2

```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 struct Node
5 {
6     int data;
7     struct Node *left;
8     struct Node *right;
9 };
10 Node(int val) {
11     data = val;
12     left = right = NULL;
13 }
14 };
15
16 Node* buildTree(string str)
17 {
18     if(str.length() == 0 || str[0] == 'N')
19         return NULL;
20
21     vector<string> ip;
22
23     istringstream iss(str);
24     for(string str; iss >> str; )
25         ip.push_back(str);
26
27     Node *root = new Node(stoi(ip[0]));
28
29     queue<Node*> queue;
30     queue.push(root);
31
32     int i = 1;
33     while(!queue.empty() && i < ip.size()) {
34         Node* currNode = queue.front();
35         queue.pop();
36
37         string currVal = ip[i];
38
39         if(currVal != "N") {
40             currNode->left = new Node(stoi(currVal));
41         }
42     }
43 }
```

Answer : 1 2 3 4 N N 5

Question - 8 :

The screenshot shows a web browser with multiple tabs. The active tab is 'check.pcc.events/next'. The page displays a coding problem titled '10 Medium'. The problem description is as follows:

Given an alphanumeric string s of lower, upper case English letters and numbers.

A good string is a string which doesn't have **two adjacent characters** $s[i]$ and $s[i + 1]$ where:

- $0 \leq i \leq s.length - 2$
- $s[i]$ is a lower-case letter and $s[i + 1]$ is the same letter but in upper-case or **vice-versa**.

To make the string good, you can choose **two adjacent** characters that make the string bad and remove them. You can keep doing this until the string becomes good.

Return *the string* after making it good. The answer is guaranteed to be unique under the given constraints.

Notice that an empty string is also good.

Input Guide :

On the right side of the browser, there is a code editor showing the following C++ code:

```
1 #include<iostream>
2 #include<vector>
3 #include<cstring>
4 #include<stack>
5 #include<algorithm>
6 using namespace std;
7
8 class Solution {
9 public:
10     string makeGood(string s) {
11         stack<int>st;
12         int n=s.size();
13         for(int i=0;i<n;i++){
14             if(st.size()==0){
15                 st.push(s[i]);
16             }
17             else{
18                 if(abs(s[i]-st.top())==32){
19                     st.top();
20                 }
21                 else{
22                     st.push(s[i]);
23                 }
24             }
25         }
26         string ans="";
27         while(!st.empty()){
28             ans.push_back(char(st.top()));
29             st.pop();
30         }
31         reverse(ans.begin(),ans.end());
32         return ans;
33     }
34 };
35
36 int main(){
37     string s;
38     cin>>s;
39     Solution ob;
40     cout<<ob.makeGood(s);
41     return 0;
42 }
```

The browser's taskbar at the bottom shows the system clock as 03:00 on 09-05-2022, with a temperature of 28°C and a 'Haze' weather condition.

Answer : "0Pabc"

Question - 9 :

The screenshot shows a web browser window with a coding competition interface. The problem is titled "13Hard" and asks to construct sentences from a string `s` and a dictionary `wordDict`. The input guide provides an example: `applepine` (main string), `4` (size of string dictionary), and `pine apple pen we` (space separated strings for dictionary). The solution is a C++ program using a recursive helper function to find all possible sentences.

13Hard

Given a string `s` and a dictionary of strings `wordDict`, add spaces in `s` to construct a sentence where each word is a valid dictionary word. Return all such possible sentences in **any order**.

Note that the same word in the dictionary may be reused multiple times in the segmentation.

Input Guide :
Insert you answer as a string and dictionary of string in the first line insert string and in next line insert length of dictionary (no of string in dictionary) then in the next line insert string in the dictionary in space separated manner like :

`applepine` (main string)
`4` (size of string dictionary)
`pine apple pen we` (space separated strings for dictionary).

```
1 #include<iostream>
2 #include<vector>
3 #include<cstring>
4 #include<stack>
5 #include<algorithm>
6 #include<map>
7 #include<bits/stdc++.h>
8 using namespace std;
9
10 class Solution {
11 public:
12     unordered_map<string,bool>map;
13     bool valid(string s){
14         if(map[s]){
15             return true;
16         }
17         return false;
18     }
19     vector<string>ans;
20     void helper(string&s,int i,string temp){
21         if(i==s.size()){
22             ans.push_back(temp);
23             return ;
24         }
25         for(int j=i;j<s.size();j++){
26             if(valid(s.substr(i,j-i+1))){
27                 if(i!=0){
28                     temp+=" ";
29                 }
30                 helper(s,j+1,temp+s.substr(i,j-i+1));
31             }
32         }
33         return ;
34     }
35     vector<string> wordBreak(string s, vector<string>& wordDict) {
36         for(int i=0;i<wordDict.size();i++){
37             map[wordDict[i]]=true;
38         }
39     }
```

Answer :

`s` : "pineapplepenapple"

Dictionary:

["apple","pen","applepen","pine","pineapple"]

Question - 10 :

The screenshot shows a web browser window with a coding competition interface. On the left, a sidebar displays a timer with '00 Days', '01 Hours', '29 Minutes', and '52 Seconds'. The main area contains a problem statement for a 'Medium' difficulty question. The problem involves an array 'prices' and a special discount rule. The input guide specifies the format for the array. On the right, a C++ code editor shows a solution using a stack to find the minimum price to the right of each item.

15 Medium

Given the array `prices` where `prices[i]` is the price of the `i`th item in a shop. There is a special discount for items in the shop, if you buy the `i`th item, then you will receive a discount equivalent to `prices[j]` where `j` is the **minimum** index such that `j > i` and `prices[j] <= prices[i]`, otherwise, you will not receive any discount at all.

Return an array where the `i`th element is the final price you will pay for the `i`th item of the shop considering the special discount.

Input Guide :
Insert you answer (which will be array)in the first line we have to insert length of array and in next line insert array as space separated values like
4
1 6 8 4 .

```
1 #include<iostream>
2 #include<vector>
3 #include<cstring>
4 #include<stack>
5 #include<algorithm>
6 #include<bits/stdc++.h>
7 using namespace std;
8
9 class Solution {
10 public:
11     vector<int> finalPrices(vector<int>& p) {
12         int n=p.size();
13         stack<int> s;
14         vector<int> d (n,-1);
15         for(int i=0;i<p.size();i++){
16             if(s.size()==0){
17                 s.push(i);
18             }
19             else{
20                 if(p[s.top()]>p[i]){
21                     while(!s.empty()&&p[s.top()]>p[i]){
22                         d[s.top()]=i;
23                         s.pop();
24                     }
25                 }
26                 s.push(i);
27             }
28         }
29         vector<int> v;
30         for(int i=0;i<n;i++){
31             if(d[i]!=-1){
32                 v.push_back(p[i]-p[d[i]]);
33             }
34             else{
35                 v.push_back(p[i]);
36             }
37         }
38         return v;
39     }
40 };
```

Answer : [1,2,3,4,5]

Question - 11 :

The screenshot shows a web browser window with a coding problem interface. The problem is titled "5Hard" and is categorized as "Hard". The problem description states: "A parentheses string is a **non-empty** string consisting only of '(' and ')'. It is valid if **any** of the following conditions is **true**:"

- It is ().
- It can be written as AB (A concatenated with B), where A and B are valid parentheses strings.
- It can be written as (A), where A is a valid parentheses string.

You are given a parentheses string *s* and a string *locked*, both of length *n*. *locked* is a binary string consisting only of '0's and '1's. For **each** index *i* of *locked*,

- If *locked[i]* is '1', you **cannot** change *s[i]*.
- But if *locked[i]* is '0', you **can** change *s[i]* to either '(' or ')'

Return true if you can make *s* a valid parentheses string. Otherwise, return false.

Input Guide :
Insert the string *s* in first line without quotes then in the second line

The solution code is written in C++ and is as follows:

```
1 #include<iostream>
2 #include<vector>
3 #include<string>
4 #include<stack>
5 using namespace std;
6
7 class Solution {
8 public:
9     bool canBeValid(string s, string l) {
10         stack<int> st;
11         int n=s.size();
12         if(n%2!=0){
13             return false;
14         }
15         for(int i=0;i<n;i++){
16             if(s[i]=='(' && !st.empty() && s[st.top()]=='('){
17                 st.pop();
18                 continue;
19             }
20             else{
21                 st.push(i);
22             }
23         }
24         if(st.size()%2!=0){
25             return false;
26         }
27         while(!st.empty()){
28             char a=s[st.top()];
29             int i=st.top();
30             st.pop();
31             char b=s[st.top()];
32             int j=st.top();
33             if(a==b){
34                 if(a=='('){
35                     if(l[i]!='0' || l[j]!='0'){
36                         return false;
37                     }
38                 }
39                 else{
40                     if(l[j]!='0'){
```

Answer : s :"))(()())))(()" l : "0101100100"

Question - 12 :

The screenshot shows a web browser with multiple tabs. The active tab is 'check.pcc.events/next'. The page displays a coding problem titled '14 Medium'. The problem description is: 'Given n non-negative integers representing an elevation map where the width of each bar is 1, compute how much water it can trap after raining.' The input guide states: 'Insert your answer (which will be array) in the first line, insert size of array and in second line insert array in a line as space separated values like :'. The input is '8' and the array is '1 3 4 0 8 9 7 6 4.'. Example 1 shows 'Input: height = [0,1,0,2,1,0,1,3,2,1,2,1]' and 'Output: 6'. To the right of the problem description is a code editor showing a C++ solution. The code uses a stack to find the maximum water that can be trapped. The code is as follows:

```
1 #include<iostream>
2 #include<stack>
3 #include<vector>
4 #include<string>
5
6 using namespace std;
7
8 class Solution {
9 public:
10     int trap(vector<int>& h) {
11         stack<int> s;
12         int n=h.size();
13         vector<int> t(h.size(),-1);
14         for(int i=0;i<n;i++){
15             if(s.empty()){
16                 s.push(i);
17             }
18             else{
19                 if(h[i]>=h[s.top()]){
20                     while(!s.empty()&&h[i]>=h[s.top()]){
21                         t[s.top()]=-1;
22                         s.pop();
23                     }
24                     s.push(i);
25                 }
26             }
27         }
28         while(!s.empty()){
29             t[s.top()]=-1;
30             s.pop();
31         }
32         int maxm=-1;
33         int curr=0;
34         for(int i=0;i<n;i++){
35             int j=i+1;
36             for(;j<t[i];j++){
37                 curr+=abs(h[i]-h[j]);
38             }
39             i=j-1;
40         }
41     }
42 }
```

The browser's taskbar at the bottom shows the system time as 03:05 on 09-05-2022, and the temperature as 28°C with a haze.

Answer : [4,2,3]

Question - 13 :

7/7 Medium

Given a string *s*, remove duplicate letters so that every letter appears once and only once. You must make sure your result is the **smallest in lexicographical order** among all possible results.

Input Guide :
Insert you ans (which will be string) without quotes like `acdfdd` .

Example 1:
Input: `s = "bcabc"`
Output: `"abc"`

Example 2:
Input: `s = "cbacdcbc"`

```
1 #include<iostream>
2 #include<vector>
3 #include<cstring>
4 #include<stack>
5 #include<algorithm>
6 #include<unordered_map>
7 using namespace std;
8
9 string removeDuplicateLetters(string s) {
10     int n=s.size();
11     unordered_map<int,int>m;
12     for(int i=0;i<n;i++){
13         if(m.count(s[i])){
14             m[s[i]]++;
15         }
16         else{
17             m[s[i]]=1;
18         }
19     }
20     stack<char>st;
21     bool arr[26]={0};
22     for(int i=0;i<n;i++){
23         if(st.empty()){
24             m[s[i]]--;
25             st.push(s[i]);
26             continue;
27         }
28         while(!st.empty()&&s[i]<=st.top()&&m[st.top()]==0&&!arr[
29             arr[st.top()-'a']=0;
30             st.pop();
31         }
32         if(!arr[s[i]-'a']){
33             st.push(s[i]);
34             arr[s[i]-'a']=1;
35         }
36         m[s[i]]--;
37     }
38     string ans="";
39     while(!st.empty()){
40         ans.push_back(st.top());
41     }
```

Answer : “abaghaa”

Question - 14 :

The screenshot shows a web browser window with a coding problem titled "12 Medium". The problem asks to check if a string `s` is valid based on three rules: 1) Any left parenthesis '(' must have a corresponding right parenthesis ')'. 2) Any right parenthesis ')' must have a corresponding left parenthesis '('. 3) Left parenthesis '(' must go before the corresponding right parenthesis ')'. Additionally, '*' could be treated as a single right parenthesis ')' or a single left parenthesis '('. The input guide says to insert the answer without quotes like `()*`.

On the left, a timer shows 00 Days, 01 Hours, 29 Minutes, and 50 Seconds.

On the right, a C++ solution is provided:

```
1 #include<iostream>
2 #include<vector>
3 #include<cstring>
4 #include<stack>
5 #include<algorithm>
6 using namespace std;
7
8 class Solution {
9 public:
10     bool checkValidString(string s) {
11         stack<int>open;
12         stack<int>star;
13         int n=s.size();
14         for(int i=0;i<n;i++){
15             if(s[i]=='('){
16                 open.push(i);
17             }
18             else if(s[i]=='*'){
19                 star.push(i);
20             }
21             else{
22                 if(!open.empty()){
23                     open.pop();
24                 }
25                 else if(!star.empty()){
26                     star.pop();
27                 }
28                 else{
29                     return false;
30                 }
31             }
32         }
33         while(!open.empty()){
34             if(star.empty()){
35                 return false;
36             }
37             open.pop();
38             star.pop();
39         }
40         return true;
41     }
42 }
```

Answer : `**()()`

Question - 15 :

The screenshot shows a web browser with a Medium article on the left and a C++ code editor on the right. The Medium article is titled "11/11 Medium" and contains the following text:

You are given two **non-empty** linked lists representing two non-negative integers. The most significant digit comes first and each of their nodes contains a single digit. Add the two numbers and return the sum as a linked list.

You may assume the two numbers do not contain any leading zero, except the number 0 itself.

Input Guide :
Insert both the list (which will be added for result) in the first line enter length of list1 in the second line enter the list 1 values separated by white spaces like 1 2 3 4. in the next line enter the length of list 2 in the next line insert list 2 values separated by white spaces
example
3
1 2 3

The C++ code on the right is a solution to the problem. It defines a `Node` struct with an `int` value and a `Node*` next pointer. It includes a `buildList` function that takes a size and builds a linked list from standard input. It also includes a `printList` function to print the linked list.

```
1 #include<iostream>
2 #include<vector>
3 #include<cstring>
4 #include<stack>
5 #include<algorithm>
6 #include<unordered_map>
7 using namespace std;
8
9
10 struct Node {
11     int val;
12     struct Node* next;
13 };
14 Node(int x) {
15     val = x;
16     next = NULL;
17 }
18
19 struct Node* buildList(int size)
20 {
21     int val;
22     cin >> val;
23
24     Node* head = new Node(val);
25     Node* tail = head;
26
27     for(int i=0; i<size-1; i++)
28     {
29         cin >> val;
30         tail->next = new Node(val);
31         tail = tail->next;
32     }
33
34     return head;
35 }
36
37 void printList(Node* n)
38 {
39     while(n)
40     {
41         cout << n->val << " ";
42         n = n->next;
43     }
44 }
```

Answer :

I1 : [9,6,4,7]

I2 : [5,6,7]