

WINTER DOMAIN CAMP

NAME: Aparajita Garbyal

UID:22BCS11904

SECTION: KPIT-901-B

DATE: 24/12/2024

Q1. Balanced brackets(Easy) Sol.

```
1 //Balanced brackets (easy)
2 #include <iostream>
3 #include <stack>
4 #include <string>
5 using namespace std;
6
7 string isBalanced(string s) {
8     stack<char> st;
9
10    for (char ch : s) {
11        if (ch == '(' || ch == '{' || ch == '[') {
12            st.push(ch);
13        } else {
14            if (st.empty()) {
15                return "NO";
16            }
17            char top = st.top();
18            if ((ch == ')' && top == '(') ||
19                (ch == '}' && top == '{') ||
20                (ch == ']' && top == '[')) {
21                st.pop();
22            } else {
23                return "NO";
24            }
25        }
26    }
27    return st.empty() ? "YES" : "NO";
28 }
29 int main() {
30     int n;
31     cin >> n;
32     cin.ignore();
33     while (n--) {
34         string s;
35         getline(cin, s);
36         cout << isBalanced(s) << endl;
37     }
38     return 0;
39 }
```

OUTPUT:

```
3
()
YES
([
NO
][{}]()
NO
```

Q2. Reverse a queue(medium)

Sol.

```

1 //Reverse a queue(medium).
2 #include <iostream>
3 #include <queue>
4 using namespace std;
5 void reverseQueue(queue<int>& q) {
6     if (q.empty()) {
7         return;
8     }
9     int front = q.front();
10    q.pop();
11    reverseQueue(q);
12    q.push(front);
13 }
14 void printQueue(queue<int> q) {
15     while (!q.empty()) {
16         cout << q.front() << " ";
17         q.pop();
18     }
19     cout << endl;
20 }
21 int main() {
22     // Example 1
23     queue<int> q1;
24     int arr1[] = {5, 24, 9, 6, 8, 4, 1, 8, 3, 6};
25     for (int num : arr1) {
26         q1.push(num);
27     }
28
29     cout << "Original Queue: ";
30     printQueue(q1);
31
32     reverseQueue(q1);
33
34     cout << "Reversed Queue: ";
35     printQueue(q1);
36
37     // Example 2
38     queue<int> q2;
39
40     int arr2[] = {8, 7, 2, 5, 1};
41     for (int num : arr2) {
42         q2.push(num);
43     }
44
45     cout << "Original Queue: ";
46     printQueue(q2);
47
48     reverseQueue(q2);
49
50     cout << "Reversed Queue: ";
51     printQueue(q2);
52
53     return 0;
54 }

```

OUTPUT:

```

Original Queue: 5 24 9 6 8 4 1 8 3 6
Reversed Queue: 6 3 8 1 4 8 6 9 24 5
Original Queue: 8 7 2 5 1
Reversed Queue: 1 5 2 7 8

```

Q3. Balanced parenthesis scoring (medium) Sol.

```
1 //balanced paranthesis scoring
2 #include <iostream>
3 #include <stack>
4 #include <string>
5 using namespace std;
6
7 int scoreOfParentheses(string s) {
8     stack<int> st;
9     st.push(0); // Initialize stack with a base score of 0
10
11     for (char c : s) {
12         if (c == '(') {
13             st.push(0); // Push a new frame for an inner score
14         } else {
15             int innerScore = st.top();
16             st.pop();
17             int outerScore = st.top();
18             st.pop();
19             int currentScore = outerScore + max(2 * innerScore, 1);
20             st.push(currentScore); // Update the score in the stack
21         }
22     }
23
24     return st.top();
25 }
26
27 int main() {
28     // Example 1
29     string s1 = "()";
30     cout << "Score of \"()\": " << scoreOfParentheses(s1) << endl;
31     // Example 2
32     string s2 = "(() )";
33     cout << "Score of \"(() )\": " << scoreOfParentheses(s2) << endl;
34     // Example 3
35     string s3 = "()()";
36     cout << "Score of \"()()\": " << scoreOfParentheses(s3) << endl;
37     return 0;
38 }
```

OUTPUT:

```
Score of "()": 1
Score of "(() )": 2
Score of "()()": 2
```

Q4. Variation game of zuma(hard)

Sol.

```

1  //Variation game of zuma(hard).
2  #include <iostream>
3  #include <string>
4  #include <unordered_map>
5  #include <algorithm>
6  #include <climits>
7  using namespace std;
8
9  // Function to remove consecutive groups of three or more balls
10 string removeConsecutive(string board) {
11     int n = board.size();
12     bool reduced = true;
13
14     while (reduced) {
15         reduced = false;
16         int i = 0;
17
18         while (i < n) {
19             int j = i;
20             while (j < n && board[j] == board[i]) {
21                 j++;
22             }
23
24             if (j - i >= 3) {
25                 board = board.substr(0, i) + board.substr(j);
26                 reduced = true;
27                 n = board.size();
28                 break;
29             }
30
31             i = j;
32         }
33     }
34
35     return board;
36 }
37
38 // Helper function for DFS

```

```

39 int dfs(string board, unordered_map<char, int>& hand) {
40     board = removeConsecutive(board);
41     if (board.empty()) return 0;
42
43     int ans = INT_MAX;
44     int n = board.size();
45
46     for (int i = 0; i < n; i++) {
47         for (auto& [color, count] : hand) {
48             if (count <= 0) continue;
49
50             string newBoard = board.substr(0, i) + color + board.substr(i);
51             hand[color]--;
52             int temp = dfs(newBoard, hand);
53             if (temp != -1) {
54                 ans = min(ans, temp + 1);
55             }
56             hand[color]++;
57         }
58     }
59
60     return ans == INT_MAX ? -1 : ans;
61 }
62
63 // Main function to calculate the minimum number of balls
64 int findMinInsertions(string board, string hand) {
65     unordered_map<char, int> handCount;
66     for (char c : hand) {
67         handCount[c]++;
68     }
69
70     return dfs(board, handCount);
71 }
72
73 int main() {
74     // Example 1
75     string board1 = "WRRBBW";
76     string hand1 = "RB";

```



```

77     cout << "Minimum insertions for \"WRRBBW\": " << findMinInsertions(board1, hand1) << endl;
78
79     // Example 2
80     string board2 = "WRRBBBW";
81     string hand2 = "WRBRW";
82     cout << "Minimum insertions for \"WRRBBBW\": " << findMinInsertions(board2, hand2) << endl;
83
84     // Example 3
85     string board3 = "G";
86     string hand3 = "GGGGG";
87     cout << "Minimum insertions for \"G\": " << findMinInsertions(board3, hand3) << endl;
88
89     return 0;
90 }

```

Output:

```

Minimum insertions for "WRRBBW": -1
Minimum insertions for "WRRBBBW": 2
Minimum insertions for "G": 2

...Program finished with exit code 0

```

Q5.Poisonous plant.(very hard)

Sol.

```

1  #include <iostream>
2  #include <vector>
3  #include <stack>
4  #include <algorithm>
5  using namespace std;
6
7  int poisonousPlants(vector<int>& p) {
8      int n = p.size();
9      vector<int> days(n, 0); // Tracks the days each plant takes to die
10     stack<int> s; // Monotonic stack for indices
11     int maxDays = 0;
12
13     for (int i = 0; i < n; i++) {
14         while (!s.empty() && p[s.top()] >= p[i]) {
15             s.pop();
16         }
17         if (!s.empty()) {
18             days[i] = days[s.top()] + 1;
19         }
20         s.push(i);
21         maxDays = max(maxDays, days[i]);
22     }
23     return maxDays;
24 }
25
26 int main() {
27     // Example 1
28     vector<int> p1 = {3, 6, 2, 7, 5};
29     cout << "Days until no plants die for example 1: " << poisonousPlants(p1) << endl;
30     // Example 2
31     vector<int> p2 = {6, 5, 8, 4, 7, 10, 9};
32     cout << "Days until no plants die for example 2: " << poisonousPlants(p2) << endl;
33     return 0;
34 }

```

OUTPUT:

```
Days until no plants die for example 1: 1  
Days until no plants die for example 2: 2
```

```
...Program finished with exit code 0  
Press ENTER to exit console. 
```