

Level Nodes

Problem	Submissions	Leaderboard	Discussions
---------	-------------	-------------	-------------

Submitted 3 hours ago • Score: 20.00

Status: Accepted

✓	Test Case #0	✓	Test Case #1	✓	Test Case #2
✓	Test Case #3	✓	Test Case #4	✓	Test Case #5
✓	Test Case #6	✓	Test Case #7	✓	Test Case #8
✓	Test Case #9	✓	Test Case #10	✓	Test Case #11
✓	Test Case #12	✓	Test Case #13	✓	Test Case #14
✓	Test Case #15	✓	Test Case #16	✓	Test Case #17
✓	Test Case #18	✓	Test Case #19	✓	Test Case #20
✓	Test Case #21	✓	Test Case #22	✓	Test Case #23
✓	Test Case #24	✓	Test Case #25	✓	Test Case #26
✓	Test Case #27	✓	Test Case #28	✓	Test Case #29
✓	Test Case #30	✓	Test Case #31	✓	Test Case #32
✓	Test Case #33	✓	Test Case #34	✓	Test Case #35
✓	Test Case #36	✓	Test Case #37	✓	Test Case #38
✓	Test Case #39	✓	Test Case #40	✓	Test Case #41
✓	Test Case #42	✓	Test Case #43	✓	Test Case #44
✓	Test Case #45	✓	Test Case #46	✓	Test Case #47
✓	Test Case #48	✓	Test Case #49	✓	Test Case #50
✓	Test Case #51	✓	Test Case #52	✓	Test Case #53
✓	Test Case #54	✓	Test Case #55	✓	Test Case #56
✓	Test Case #57	✓	Test Case #58	✓	Test Case #59
✓	Test Case #60				

Submitted Code

Language: C++20

Open in editor

```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 class Node
```

```
5 {
6 public:
7     int val;
8     Node *left;
9     Node *right;
10    Node(int val)
11    {
12        this->val = val;
13        this->left = NULL;
14        this->right = NULL;
15    }
16 };
17
18 stack<int> st;
19 void noOfLevels(Node *root)
20 {
21     if (root == NULL)
22     {
23         return;
24     }
25
26     queue<pair<Node *, int>> q;
27     if (root)
28         q.push({root, 0});
29
30     while (!q.empty())
31     {
32         // queue theke ber korlam
33         pair<Node *, int> pr = q.front();
34         Node *f = pr.first;
35         int level = pr.second;
36         q.pop();
37
38         // jabotiyo kaj
39         st.push(level);
40
41
42         // queue te push korlam
43         if (f->left)
44             q.push({f->left, level + 1});
45         if (f->right)
46             q.push({f->right, level + 1});
47     }
48 }
49
50 Node *inputTree()
51 {
52     int val;
53     cin >> val;
54     Node *root;
55     if (val == -1)
56         root = NULL;
57     else
58         root = new Node(val);
59
60     queue<Node *> q;
61     if (root)
62         q.push(root);
63
64     while (!q.empty())
65     {
66         Node *f = q.front();
67         q.pop();
68
69         int l, r;
70         cin >> l >> r;
```

```
71     Node *myLeft, *myRight;
72
73     if (l == -1)
74         myLeft = NULL;
75     else
76         myLeft = new Node(l);
77
78     if (r == -1)
79         myRight = NULL;
80     else
81         myRight = new Node(r);
82
83     f->left = myLeft;
84     f->right = myRight;
85
86     if (f->left)
87         q.push(f->left);
88     if (f->right)
89         q.push(f->right);
90 }
91 return root;
92 }
93
94 void levelNodes(Node *root, int lvl, int mxLevel) // 10 3
95 {
96     if (root == NULL)
97     {
98         return;
99     }
100
101     queue<pair<Node *, int>> q;
102     if (root)
103         q.push({root, 0});
104
105     while (!q.empty())
106     {
107         // queue theke ber korlam
108         pair<Node *, int> pr = q.front();
109         Node *f = pr.first;
110         int level = pr.second;
111         q.pop();
112
113         // jabotiyo kaj
114         if (lvl > mxLevel)
115         {
116             cout<<"Invalid"<<endl;
117             return;
118         }
119         else if (level == lvl)
120         {
121             cout << f->val << " ";
122         }
123
124         // queue te push korlam
125         if (f->left)
126             q.push({f->left, level + 1});
127         if (f->right)
128             q.push({f->right, level + 1});
129     }
130 }
131
132 /*
133 10 20 30 40 50 -1 60
134 -1 -1 -1 -1 -1 -1
135 */
136 int main()
```

```
137 {  
138     Node *root = inputTree();  
139     int l;  
140     cin >> l;  
141  
142     noOfLevels(root);  
143     // cout<<st.top()<<endl;  
144     int highestLevel = st.top();  
145     levelNodes(root, l, highestLevel);  
146  
147  
148     return 0;  
149 }
```