



## Symmetric Tree

Easy

Accuracy: 44.96%

Submissions: 97K+

Points: 2

Unlock your coding potential - Join our Hiring Coding Contest and land your dream job!

Given a Binary Tree. Check whether it is Symmetric or not, i.e. whether the binary tree is a **Mirror image of itself** or not.

### Example 1:

Input:



Output: True

Explanation: Tree is mirror image of itself i.e. tree is symmetric

### Example 2:

```
1 // } Driver Code Ends
89
90
91 class Solution{
92 public:
93     // return true/false denoting whether the tree is Symmetric or not
94     bool check(Node* r1, Node* r2){
95
96         if(r1 == NULL && r2 == NULL)
97             return true;
98         if(r1 == NULL || r2 == NULL)
99             return false;
100         if(r1->data != r2->data)
101             return false;
102
103         if(check(r1->left, r2->right) == false || check(r1->right, r2->left) == false)
104             return false;
105         return true;
106     }
107 }
108
109 bool isSymmetric(struct Node* root){
110     if(root == NULL)
111         return true;
112
113     return check(root->left, root->right);
114 }
115 };
116 // } Driver Code Ends
```





## Vertical Traversal of Binary Tree



Medium

Accuracy: 32.87%

Submissions: 147K+

Points: 4

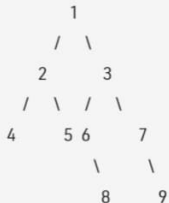
Unlock your coding potential - Join our Hiring Coding Contest and land your dream job!

Given a Binary Tree, find the vertical traversal of it starting from the leftmost level to the rightmost level.

If there are multiple nodes passing through a vertical line, then they should be printed as they appear in **level order** traversal of the tree.

### Example 1:

Input:



Output:

```
1 // } Driver Code Ends
100 class Solution
101 {
102     public:
103     //Function to find the vertical order traversal of Binary Tree.
104     vector<int> verticalOrder(Node *root)
105     {
106         vector<int> res;
107         if(root == NULL)
108             return {};
109         queue< pair<Node*, int> > q;
110         map<int, vector<int> > mp;
111
112         q.push({root, 0});
113
114         while( !q.empty() ) {
115             root = q.front().first;
116             int col = q.front().second;
117             q.pop();
118
119             mp[col].emplace_back(root->data);
120
121             if(root->left)
122                 q.push({root->left, col - 1});
123             if(root->right)
124                 q.push({root->right, col + 1});
125         }
126
127         for(auto &i : mp) {
128             res.insert(res.end(), i.second.begin(), i.second.end() );
129         }
130         return res;
131     }
132 };
133
134
```

