

DAY 6

Name – Rohit

UID – 22BCS15476

Section – 620-B

Q1. Given the root of a binary tree, return the inorder traversal of its nodes' values.

Ans

```
#include <iostream>
```

```
#include <vector>
```

```
using namespace std;
```

```
struct TreeNode {
```

```
    int val;
```

```
    TreeNode* left;
```

```
    TreeNode* right;
```

```
    TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
```

```
};
```

```
void inorderTraversal(TreeNode* root, vector<int>& result) {
```

```
    if (root == nullptr) {
```

```
        return;
```

```
    }
```

```
    inorderTraversal(root->left, result);
```

```
    result.push_back(root->val);
```

```
    inorderTraversal(root->right, result);
```

```
}
```

```
TreeNode* createExampleTree() {
```

```

TreeNode* root = new TreeNode(1);
root->right = new TreeNode(2);
root->right->left = new TreeNode(3);
return root;
}

int main() {
    TreeNode* root = createExampleTree();
    vector<int> result;
    inorderTraversal(root, result);
    cout << "Inorder traversal: ";
    for (int val : result) {
        cout << val << " ";
    }
    cout << endl;
    return 0;
}

```

Output

```

Inorder traversal: 1 3 2

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

```

Q2. Given the root of a complete binary tree, return the number of the nodes in the tree.

Ans

```

#include <iostream>

using namespace std;

struct TreeNode {
    int val;

    TreeNode* left;
    TreeNode* right;

    TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
};

int countNodes(TreeNode* root) {
    if (root == nullptr) {
        return 0;
    }

    return 1 + countNodes(root->left) + countNodes(root->right);
}

TreeNode* createExampleTree() {
    TreeNode* root = new TreeNode(1);
    root->left = new TreeNode(2);
    root->right = new TreeNode(3);
    root->left->left = new TreeNode(4);
    return root;
}

int main() {
    TreeNode* root = createExampleTree();
    int nodeCount = countNodes(root);

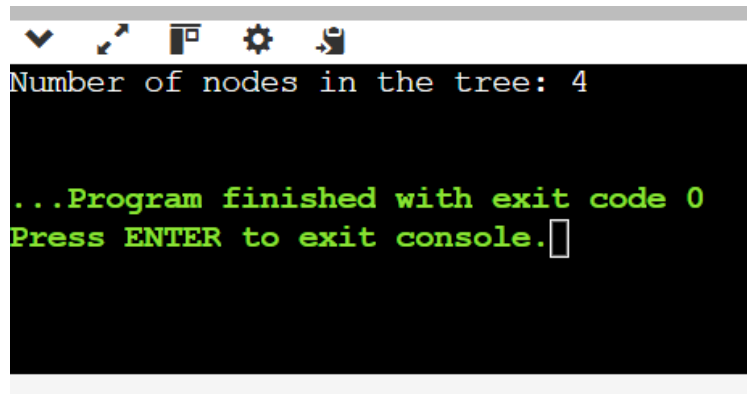
    cout << "Number of nodes in the tree: " << nodeCount << endl; // Output: 6

    return 0;
}

```

```
}
```

Output



```
Number of nodes in the tree: 4

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

Q3. Symmetric tree

Ans

```
#include <iostream>

using namespace std;

struct TreeNode {
    int val;
    TreeNode* left;
    TreeNode* right;
    TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
};

int isMirror(TreeNode* left, TreeNode* right) {
    if (left == nullptr && right == nullptr) {
        return true;
    }
    if (left == nullptr || right == nullptr) {
        return false;
    }
}
```

```

    return (left->val == right->val) && isMirror(left->left, right->right) &&
isMirror(left->right, right->left);
}

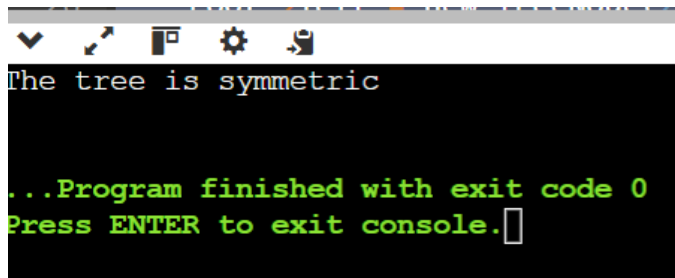
int isSymmetric(TreeNode* root) {
    if (root == nullptr) {
        return true;
    }
    return isMirror(root->left, root->right);
}

TreeNode* createExampleTree() {
    TreeNode* root = new TreeNode(1);
    root->left = new TreeNode(2);
    root->right = new TreeNode(2);
    root->left->left = new TreeNode(3);
    root->left->right = new TreeNode(4);
    root->right->left = new TreeNode(4);
    root->right->right = new TreeNode(3);
    return root;
}

int main() {
    TreeNode* root = createExampleTree();
    bool symmetric = isSymmetric(root);
    cout << "The tree is " << (symmetric ? "symmetric" : "not symmetric") <<
endl;
    return 0;
}

```

Output

A screenshot of a console window with a black background and green text. The text reads: "The tree is symmetric", followed by "...Program finished with exit code 0", and "Press ENTER to exit console." with a cursor at the end. The window has a standard OS title bar and a toolbar with icons for back, forward, search, and other navigation functions.

```
The tree is symmetric

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

Q4. Inorder traversal inverting

Ans

```
#include <iostream>

using namespace std;

struct TreeNode {
    int val;
    TreeNode* left;
    TreeNode* right;
    TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
};

TreeNode* invertTree(TreeNode* root) {
    if (root == nullptr) {
        return root;
    }
    TreeNode* temp = root->left;
    root->left = root->right;
    root->right = temp;
    invertTree(root->left);
    invertTree(root->right);
    return root;
}
```

```

TreeNode* createExampleTree() {
    TreeNode* root = new TreeNode(1);
    root->left = new TreeNode(2);
    root->right = new TreeNode(3);
    root->left->left = new TreeNode(4);
    root->left->right = new TreeNode(5);
    root->right->left = new TreeNode(6);
    root->right->right = new TreeNode(7);
    return root;
}

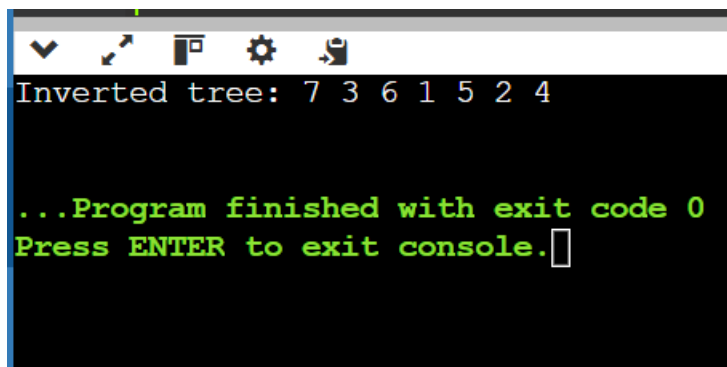
void printInOrder(TreeNode* root) {
    if (root == nullptr) {
        return;
    }
    printInOrder(root->left);
    cout << root->val << " ";
    printInOrder(root->right);
}

int main() {
    TreeNode* root = createExampleTree();
    root = invertTree(root);
    cout << "Inverted tree: ";
    printInOrder(root);
    cout << endl;
    return 0;
}

```

```
}
```

Output



```
Inverted tree: 7 3 6 1 5 2 4

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

Q5. Path sum

Ans

```
#include <iostream>
```

```
using namespace std;
```

```
struct TreeNode {
```

```
    int val;
```

```
    TreeNode* left;
```

```
    TreeNode* right;
```

```
    TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
```

```
};
```

```
bool hasPathSum(TreeNode* root, int sum) {
```

```
    if (root == nullptr) {
```

```
        return false;
```

```
    }
```

```
    if (root->left == nullptr && root->right == nullptr) {
```

```
        return (sum == root->val);
```

```
    }
```

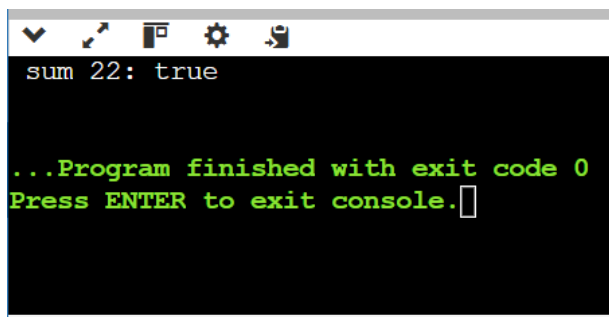


```
    return hasPathSum(root->left, sum - root->val) || hasPathSum(root->right,
sum - root->val);
}
```

```
TreeNode* createExampleTree() {
    TreeNode* root = new TreeNode(5);
    root->left = new TreeNode(4);
    root->right = new TreeNode(8);
    root->left->left = new TreeNode(11);
    root->left->left->left = new TreeNode(7);
    root->left->left->right = new TreeNode(2);
    root->right->left = new TreeNode(13);
    root->right->right = new TreeNode(4);
    root->right->right->right = new TreeNode(1);
    return root;
}
```

```
int main() {
    TreeNode* root = createExampleTree();
    int sum = 22;
    bool result = hasPathSum(root, sum);
    cout << " sum " << sum << ": " << (result ? "true" : "false") << endl;
    return 0;
}
```

} Output

A screenshot of a console window with a black background and green text. The window has a standard OS title bar with icons for minimize, maximize, and close. The text in the console reads: "sum 22: true" on the first line, followed by "...Program finished with exit code 0" on the second line, and "Press ENTER to exit console." on the third line, with a cursor at the end of the last line.

```
sum 22: true
...Program finished with exit code 0
Press ENTER to exit console.
```

Q6. Constructed tree (in-order traversal)

Ans

```
#include <iostream>
```

```
#include <vector>
```

```
#include <unordered_map>
```

```
using namespace std;
```

```
struct TreeNode {
```

```
    int val;
```

```
    TreeNode* left;
```

```
    TreeNode* right;
```

```
    TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
```

```
};
```

```
TreeNode* buildTreeHelper(vector<int>& inorder, vector<int>& postorder, int  
inStart, int inEnd, int& postIndex, unordered_map<int, int>& inMap) {
```

```
    if (inStart > inEnd) {
```

```
        return nullptr;
```

```
    }
```

```
    int rootVal = postorder[postIndex--];
```

```
    TreeNode* root = new TreeNode(rootVal);
```

```
    int inIndex = inMap[rootVal];
```

```
    root->right = buildTreeHelper(inorder, postorder, inIndex + 1, inEnd,  
postIndex, inMap);
```

```
    root->left = buildTreeHelper(inorder, postorder, inStart, inIndex - 1,  
postIndex, inMap);
```

```
    return root;
```

```
}
```

```

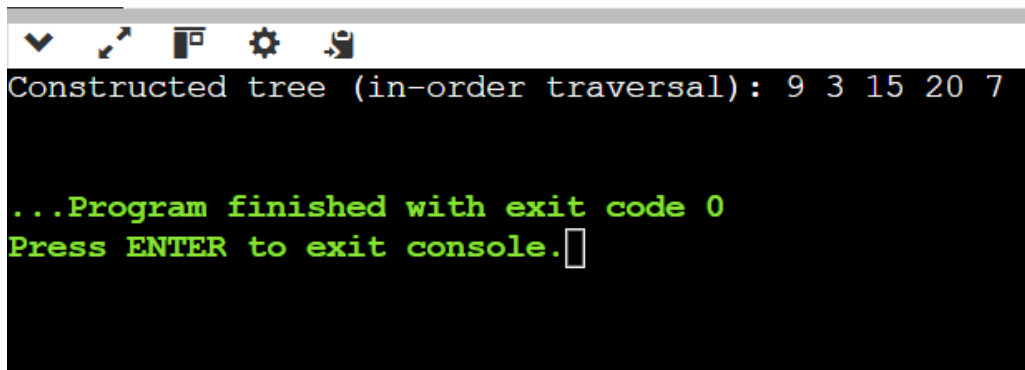
TreeNode* buildTree(vector<int>& inorder, vector<int>& postorder) {
    unordered_map<int, int> inMap;
    for (int i = 0; i < inorder.size(); ++i) {
        inMap[inorder[i]] = i;
    }
    int postIndex = postorder.size() - 1;
    return buildTreeHelper(inorder, postorder, 0, inorder.size() - 1, postIndex,
inMap);
}

void printInOrder(TreeNode* root) {
    if (root == nullptr) {
        return;
    }
    printInOrder(root->left);
    cout << root->val << " ";
    printInOrder(root->right);
}

int main() {
    vector<int> inorder = {9, 3, 15, 20, 7};
    vector<int> postorder = {9, 15, 7, 20, 3};
    TreeNode* root = buildTree(inorder, postorder);
    cout << "Constructed tree (in-order traversal): ";
    printInOrder(root);
    cout << endl;
    return 0;
}

```

Output



```
Constructed tree (in-order traversal): 9 3 15 20 7

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

Q7. Sum of all root leaf no.

Ans

```
#include <iostream>

using namespace std;

struct TreeNode {
    int val;
    TreeNode* left;
    TreeNode* right;
    TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
};

int sumNumbersHelper(TreeNode* root, int currentSum) {
    if (root == nullptr) {
        return 0;
    }
    currentSum = currentSum * 10 + root->val;
    if (root->left == nullptr && root->right == nullptr) {
        return currentSum;
    }
    return sumNumbersHelper(root->left, currentSum) +
        sumNumbersHelper(root->right, currentSum);
}
```

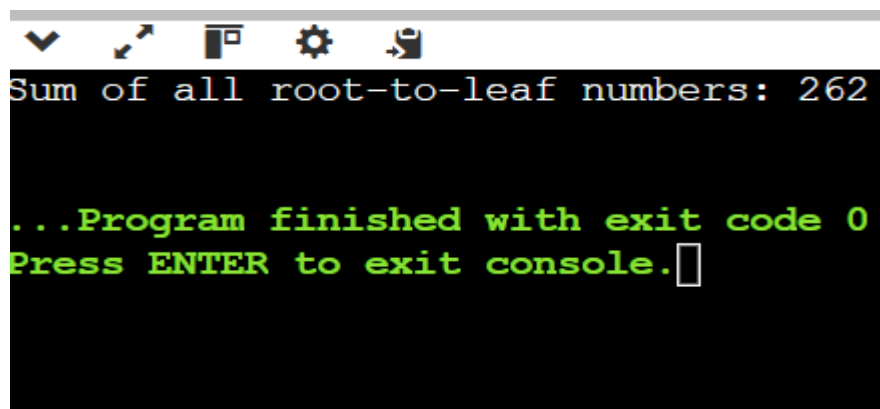
```
}
```

```
int sumNumbers(TreeNode* root) {  
    return sumNumbersHelper(root, 0);  
}
```

```
TreeNode* createExampleTree() {  
    TreeNode* root = new TreeNode(1);  
    root->left = new TreeNode(2);  
    root->right = new TreeNode(3);  
    root->left->left = new TreeNode(4);  
    root->left->right = new TreeNode(5);  
    return root;  
}
```

```
int main() {  
    TreeNode* root = createExampleTree();  
    int result = sumNumbers(root);  
    cout << "Sum of all root-to-leaf numbers: " << result << endl;  
    return 0;  
}
```

Output

A screenshot of a console window with a dark background. The title bar is light gray and contains several icons: a checkmark, a cursor, a square, a gear, and a document. The console text is as follows:
Sum of all root-to-leaf numbers: 262

...Program finished with exit code 0
Press ENTER to exit console.
A white cursor is visible at the end of the last line.

Q8. Zigzag level order traversal

Ans

```
#include <iostream>

#include <vector>

#include <deque>

using namespace std;

struct TreeNode {

    int val;

    TreeNode* left;

    TreeNode* right;

    TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}

};

vector<vector<int>> zigzagLevelOrder(TreeNode* root) {

    vector<vector<int>> result;

    if (root == nullptr) {

        return result;

    }

    deque<TreeNode*> dq;

    dq.push_back(root);

    bool leftToRight = true;

    while (!dq.empty()) {

        int levelSize = dq.size();

        vector<int> level(levelSize);

        for (int i = 0; i < levelSize; ++i) {

            TreeNode* node;

            if (leftToRight) {
```

```

        node = dq.front();
        dq.pop_front();
        level[i] = node->val;
        if (node->left) dq.push_back(node->left);
        if (node->right) dq.push_back(node->right);
    } else {
        node = dq.back();
        dq.pop_back();
        level[i] = node->val;
        if (node->right) dq.push_front(node->right);
        if (node->left) dq.push_front(node->left);
    }
}

result.push_back(level);

leftToRight = !leftToRight;
}

return result;
}

TreeNode* createExampleTree() {
    TreeNode* root = new TreeNode(3);
    root->left = new TreeNode(9);
    root->right = new TreeNode(20);
    root->right->left = new TreeNode(15);
    root->right->right = new TreeNode(7);
    return root;
}

```

```

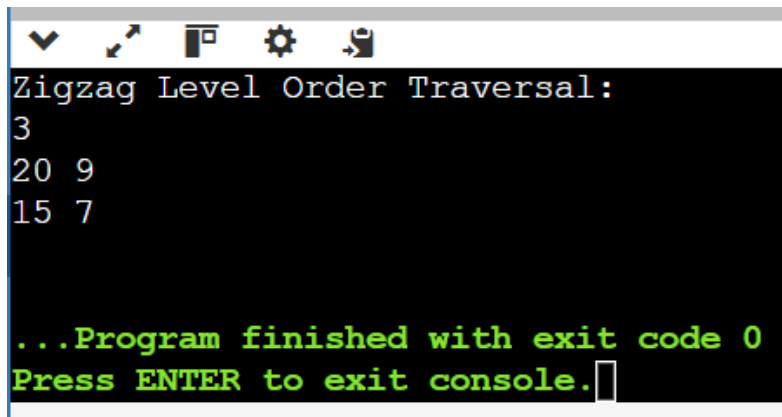
int main() {
    TreeNode* root = createExampleTree();
    vector<vector<int>> result = zigzagLevelOrder(root);

    cout << "Zigzag Level Order Traversal: " << endl;
    for (const auto& level : result) {
        for (int val : level) {
            cout << val << " ";
        }
        cout << endl;
    }

    return 0;
}

```

Output



```

Zigzag Level Order Traversal:
3
20 9
15 7

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

```

Q9. Given a binary search tree (BST), write a function to find the kth smallest element in the tree.

Ans

```
#include <iostream>
```



```

using namespace std;

struct TreeNode {
    int val;
    TreeNode* left;
    TreeNode* right;
    TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
};

void inorder(TreeNode* root, int& k, int& result) {
    if (root == nullptr) {
        return;
    }

    inorder(root->left, k, result);

    if (--k == 0) {
        result = root->val;
        return;
    }

    inorder(root->right, k, result);
}

int kthSmallest(TreeNode* root, int k) {
    int result = -1;
    inorder(root, k, result);
    return result;
}

```

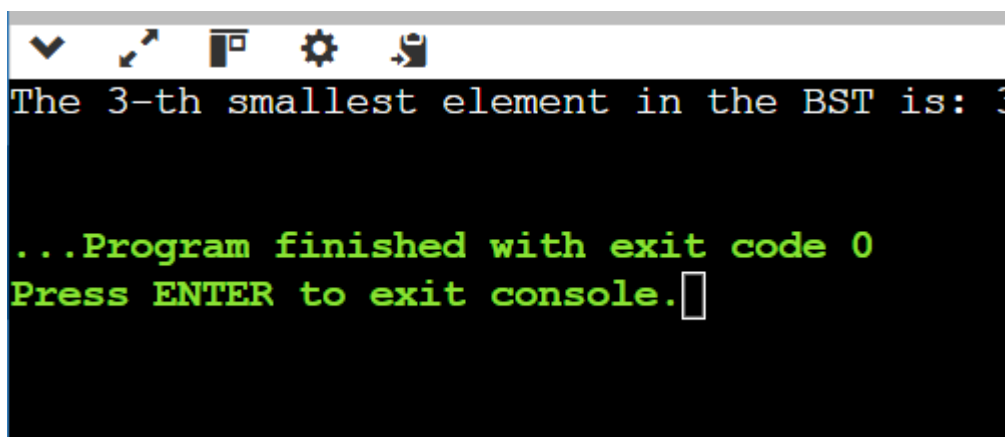
```

TreeNode* createExampleTree() {
    TreeNode* root = new TreeNode(5);
    root->left = new TreeNode(3);
    root->right = new TreeNode(6);
    root->left->left = new TreeNode(2);
    root->left->right = new TreeNode(4);
    root->left->left->left = new TreeNode(1);
    return root;
}

int main() {
    TreeNode* root = createExampleTree();
    int k = 3;
    int result = kthSmallest(root, k);
    cout << "The " << k << "-th smallest element in the BST is: " << result << endl;
    return 0;
}

```

Output



```

The 3-th smallest element in the BST is: 3
...Program finished with exit code 0
Press ENTER to exit console.

```

Q10. Sum of all nodes given the root of binary tree

Ans

```

#include <iostream>

using namespace std;

struct TreeNode {
    int val;

    TreeNode* left;
    TreeNode* right;

    TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
};

int sumOfNodes(TreeNode* root) {
    if (root == nullptr) {
        return 0;
    }

    return root->val + sumOfNodes(root->left) + sumOfNodes(root->right);
}

TreeNode* createExampleTree() {
    TreeNode* root = new TreeNode(1);
    root->left = new TreeNode(2);
    root->right = new TreeNode(3);
    root->left->right = new TreeNode(5);
    root->right->right = new TreeNode(6);
    return root;
}

int main() {
    TreeNode* root = createExampleTree();

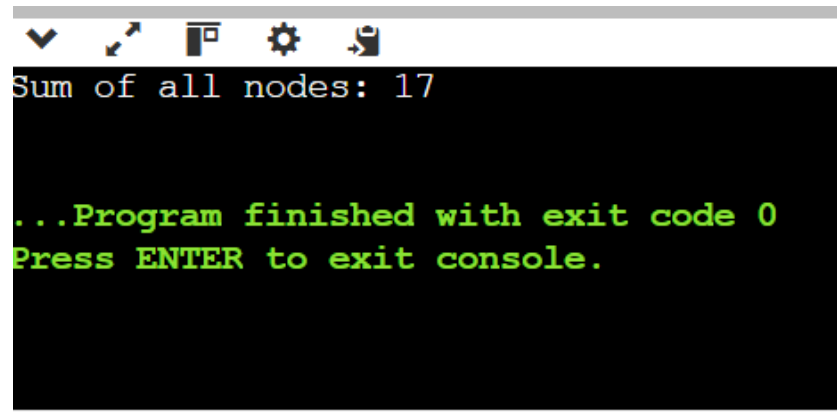
    int sum = sumOfNodes(root);

    cout << "Sum of all nodes: " << sum << endl;
}

```

```
    return 0;  
}
```

Output

A screenshot of a terminal window. The window has a title bar with standard icons (minimize, maximize, close, settings, and a terminal icon). The terminal background is black with white text. The first line of output is "Sum of all nodes: 17". The second line is "...Program finished with exit code 0". The third line is "Press ENTER to exit console.".

```
Sum of all nodes: 17  
  
...Program finished with exit code 0  
Press ENTER to exit console.
```