Name: Keshav Kumar Soni

UID: 22BCS12587

Section: KPIT_901/B

DAY 6

1. Same Tree

Two binary trees are considered the same if they are structurally identical, and the nodes have the same value.

Code:
```cpp
#include <iostream>
using namespace std;

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

bool isSameTree(TreeNode* p, TreeNode* q) {
    if (!p && !q) return true;
    if (!p || !q || p->val != q->val) return false;
    return isSameTree(p->left, q->left) && isSameTree(p->right, q->right);
}

int main() {
    TreeNode* p = new TreeNode(1);
    p->left = new TreeNode(2);    p->right = new TreeNode(3);

    TreeNode* q = new TreeNode(1);
    q->left = new TreeNode(2);    q->right = new TreeNode(3);

    cout << (isSameTree(p, q) ? "true" : "false") << endl; // Output: true
    return 0;
}
```
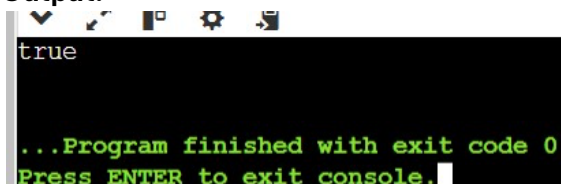
Output:

2.

Symmetric Tree

Code: #include <iostream> using
namespace std;

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

bool isMirror(TreeNode* t1, TreeNode* t2) {
if (!t1 && !t2) return true;
    if (!t1 || !t2 || t1->val != t2->val) return false;
    return isMirror(t1->left, t2->right) && isMirror(t1->right, t2->left);
}

bool isSymmetric(TreeNode* root) {
return isMirror(root, root);
}

int main() {
    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);

    cout << (isSymmetric(root) ? "true" : "false") << endl; // Output: true
return 0;
}
```
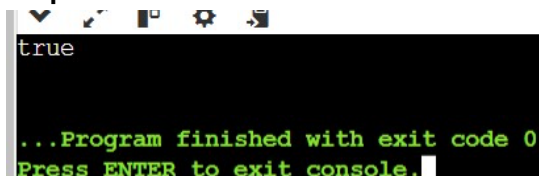
Output:

3.

Invert Binary Tree

Code:
```cpp
#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) return nullptr;    swap(root->left,
root->right);    invertTree(root->left);
invertTree(root->right);
    return root;
}

void printTree(TreeNode* root) {
if (!root) return;    cout << root-
>val << " ";    printTree(root-
>left);
    printTree(root->right);
}

int main() {
    TreeNode* root = new TreeNode(4);
root->left = new TreeNode(2);    root->right
= new TreeNode(7);    root->left->left =
new TreeNode(1);    root->left->right = new
TreeNode(3);    root->right->left = new
TreeNode(6);
    root->right->right = new TreeNode(9);

    root = invertTree(root);

    printTree(root); // Output: 4 7 9 6 2 3 1
    return 0;
}
```
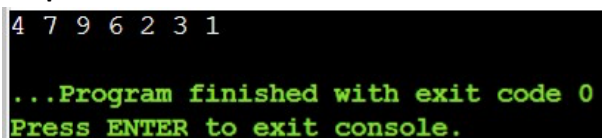**Output:**

Leaf Nodes of a Binary Tree

4.

Code: #include
<iostream>
using namespace std;
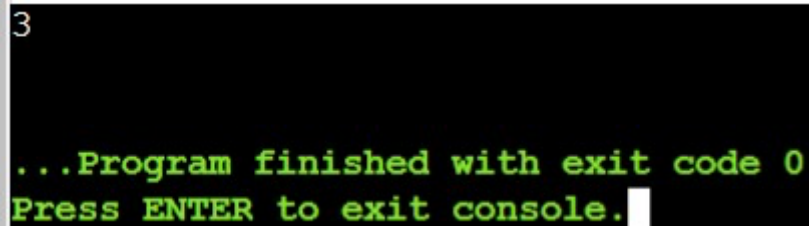
```cpp
struct TreeNode {
    int val;
    TreeNode* left;
    TreeNode* right;
    TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
};

int countLeaves(TreeNode* root) {
if (!root) return 0;
    if (!root->left && !root->right) return 1;
    return countLeaves(root->left) + countLeaves(root->right);
}

int main() {
    TreeNode* root = new TreeNode(1);    root-
>left = new TreeNode(2);    root->left->left =
new TreeNode(3);    root->left->right = new
TreeNode(4);
    root->right = new TreeNode(5);

    cout << countLeaves(root) << endl; // Output: 3
return 0;
}
```
Output:

Path Sum

5.

Code:
```cpp
#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 targetSum) {
    if (!root) return false;
    if (!root->left && !root->right) return root->val == targetSum;
    return hasPathSum(root->left, targetSum - root->val) || hasPathSum(root->right,
targetSum - root->val);
}

int main() {
    TreeNode* root = new TreeNode(5);    root-
>left = new TreeNode(4);    root->left->left =
new TreeNode(11);    root->left->left->left =
new TreeNode(7);    root->left->left->right =
new TreeNode(2);    root->right = new
TreeNode(8);    root->right->left = new
TreeNode(13);    root->right->right = new
TreeNode(4);
    root->right->right->right = new TreeNode(1);

    cout << (hasPathSum(root, 22) ? "true" : "false") << endl; // Output: true
return 0;
}
```
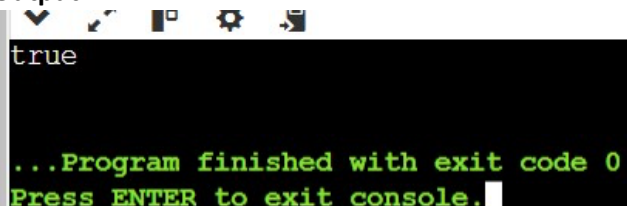**Output:**