DEPARTMENT OF

Discover. Learn. Empower.

Day 6

Student Name: Sweta singh **UID: 22BCS10664**

Branch: CSE 3rdyear **Section/Group:** KPIT-901 A

Q1. **Binary Tree Inorder Traversal.**

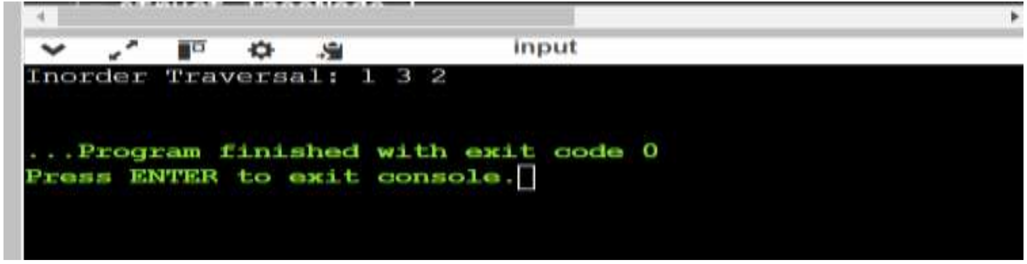
Given the root of a binary tree, return the inorder traversal of its nodes' values.

Code :

```
#include <stdio.h>
#include <stdlib.h>
struct TreeNode {
   int val;
   struct TreeNode* left;
   struct TreeNode* right;
};
struct TreeNode* newNode(int value) {
   struct TreeNode* node = (struct TreeNode*)malloc(sizeof(struct TreeNode));
   node->val = value;
   node->left = NULL;
   node->right = NULL;
   return node;
}
void inorderTraversalHelper(struct TreeNode* root) {
   if (root == NULL) {
      return;
   }
inorderTraversalHelper(root->left);
printf("%d ", root->val);
inorderTraversalHelper(root->right);
}
void inorderTraversal(struct TreeNode* root) {
   printf("Inorder Traversal: ");
   inorderTraversalHelper(root);
```

```
 printf("\n");
}
int main() {
    struct TreeNode* root = newNode(1);
    root->right = newNode(2);
    root->right->left = newNode(3);
    inorderTraversal(root);
    return 0;
}
```

Output :



```
Inorder Traversal: 1 3 2


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

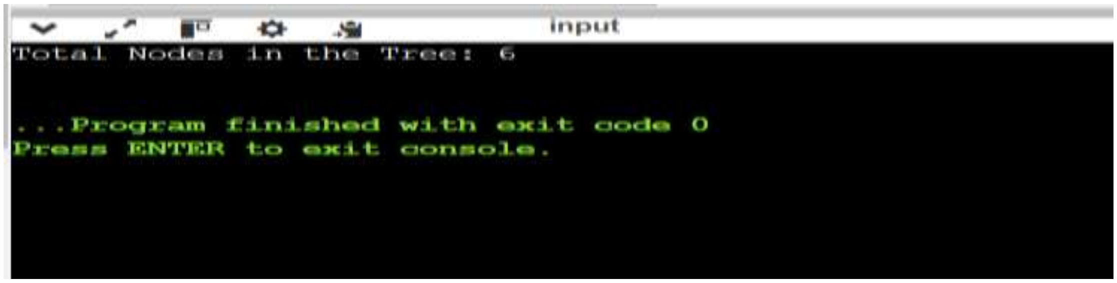
Q2. **Count Complete Tree Nodes**
Given the root of a complete binary tree, return the number of the nodes in the tree.

Code:

```
#include <stdio.h>

#include <stdlib.h>

struct TreeNode {

    int val;

    struct TreeNode* left;

    struct TreeNode* right;

};

struct TreeNode* newNode(int value) {

    struct TreeNode* node = (struct TreeNode*)malloc(sizeof(struct

TreeNode));

    node->val = value;

    node->left = NULL;

    node->right = NULL;

    return node;
```

```
int countNodes(struct TreeNode* root) {
    if (root == NULL) {
        return 0;
    }

    int leftCount = countNodes(root->left);
    int rightCount = countNodes(root->right);

    return 1 + leftCount + rightCount;
}
int main() {
    struct TreeNode* root = newNode(1);
    root->left = newNode(2);
    root->right = newNode(3);
    root->left->left = newNode(4);
    root->left->right = newNode(5);
    root->right->left = newNode(6);
    int totalNodes = countNodes(root);
    printf("Total Nodes in the Tree: %d\n", totalNodes);

    return 0;
}
```

Output:



```
Total Nodes in the Tree: 6


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

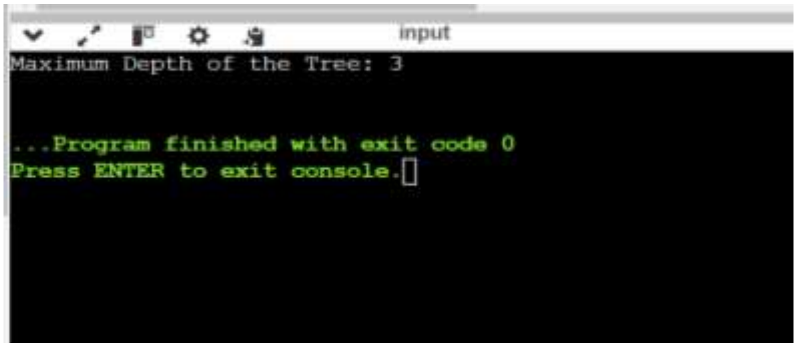
Q3. **Binary Tree - Find Maximum Depth**
A binary tree's maximum depth is the number of nodes along the longest path from the root node down to the farthest leaf node.

Code:

```
#include <stdio.h>

#include <stdlib.h>

struct TreeNode {

    int val;

    struct TreeNode* left;

    struct TreeNode* right;

};

struct TreeNode* newNode(int value) {

    struct TreeNode* node = (struct TreeNode*)malloc(sizeof(struct TreeNode));

    node->val = value;

    node->left = NULL;

    node->right = NULL;

    return node;

}

int maxDepth(struct TreeNode* root) {

    if (root == NULL) {

        return 0;

    }

    int leftDepth = maxDepth(root->left);

    int rightDepth = maxDepth(root->right);


    return 1 + (leftDepth > rightDepth ? leftDepth : rightDepth);

}
```

```
int main() {
    struct TreeNode* root = newNode(1);
    root->left = newNode(2);
    root->right = newNode(3);
    root->left->left = newNode(4);
    root->left->right = newNode(5);
    int depth = maxDepth(root);
    printf("Maximum Depth of the Tree: %d\n", depth);

    return 0;
}
```

Output:



```
Maximum Depth of the Tree: 3


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

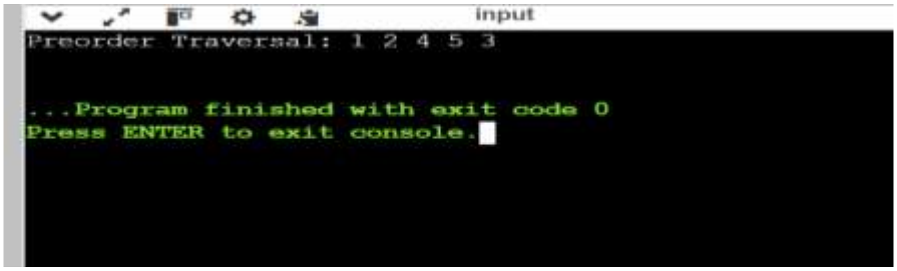
Q4. Binary Tree Preorder Traversal

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

Code:

```
#include <stdio.h>

#include <stdlib.h>

struct TreeNode {

    int val;

    struct TreeNode* left;
    struct TreeNode* right;

};

struct TreeNode* newNode(int value) {

    struct TreeNode* node = (struct TreeNode*)malloc(sizeof(struct TreeNode));

    node->val = value;

    node->left = NULL;

    node->right = NULL;

    return node;

}

void preorderTraversalHelper(struct TreeNode* root) {

    if (root == NULL) {

        return;

    }

    printf("%d ", root->val);

    preorderTraversalHelper(root->left);

    preorderTraversalHelper(root->right);

}

void preorderTraversal(struct TreeNode* root) {

    printf("Preorder Traversal: ");

    preorderTraversalHelper(root);

    printf("\n");

}

int main() {

    struct TreeNode* root = newNode(1);
    root->left = newNode(2);
    root->right = newNode(3);
    root->left->left = newNode(4);
    root->left->right = newNode(5);
    preorderTraversal(root);
return 0;
}
```

Output :



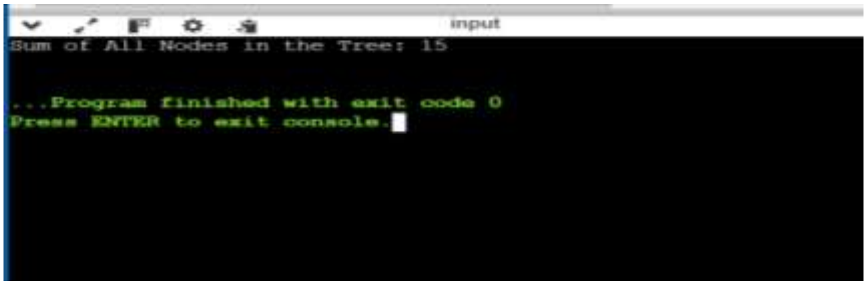
```
Preorder Traversal: 1 2 4 5 3

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

Q5. **Binary Tree - Sum of All Nodes**
Given the root of a binary tree, you need to find the sum of all the node values in the binary tree.

Code:

```
#include <stdio.h>
#include <stdlib.h>
struct TreeNode {
    int val;
    struct TreeNode* left;
    struct TreeNode* right;
};
struct TreeNode* newNode(int value) {
    struct TreeNode* node = (struct TreeNode*)malloc(sizeof(struct TreeNode));
    node->val = value;
    node->left = NULL;
    node->right = NULL;
    return node;
}
int sumOfNodes(struct TreeNode* root) {
    if (root == NULL) {
        return 0;
    }
    int leftSum = sumOfNodes(root->left);
    int rightSum = sumOfNodes(root->right);

    return root->val + leftSum + rightSum;
}
int main() {
    struct TreeNode* root = newNode(1);
    root->left = newNode(2);
    root->right = newNode(3);
    root->left->left = newNode(4);
    root->left->right = newNode(5);
    int totalSum = sumOfNodes(root);
    printf("Sum of All Nodes in the Tree: %d\n", totalSum);
    return 0;
}
```

Output :



```
Sum of All Nodes in the Tree: 15


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