Winter Winning Camp 1

Name : Sagar Papta **UID : 22BCS12131**
Branch: BE CSE **Section : ML-904**

Code 1:

```cpp
#include <iostream>
#include <queue>
using namespace std;

struct TreeNode {
    int val;
    TreeNode* left;
    TreeNode* right;
    TreeNode(int x) : val(x), left(NULL), right(NULL) {}
};
void inorderTraversal(TreeNode* root) {
    if (root == NULL) return;
    inorderTraversal(root->left);
    cout << root->val << " ";
    inorderTraversal(root->right);
}
TreeNode* insertLevelOrder(int arr[], int n, int i) {
    TreeNode* root = NULL;
    if (i < n) {
        root = new TreeNode(arr[i]);
        root->left = insertLevelOrder(arr, n, 2 * i + 1);
        root->right = insertLevelOrder(arr, n, 2 * i + 2);
    }
    return root;
}
int main() {
    int n;
    cout << "Enter the number of elements in the array (including null as -1): ";
    cin >> n;

    int* arr = new int[n];
    cout << "Enter the elements of the array: ";
    for (int i = 0; i < n; i++) {
        cin >> arr[i];
    }

    TreeNode* root = insertLevelOrder(arr, n, 0);
    cout << "Inorder Traversal: ";
    inorderTraversal(root);
```

```
    cout << endl;
    delete[] arr;
    return 0;
}
```

OUTPUT:
Enter the number of elements in the array (including null as -1): 5
Enter the elements of the array: 1 -1 2 3 -1
Inorder Traversal: 3 -1 -1 1 2

Code2:

```
#include <iostream>
#include <cmath>
using namespace std;

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

int getDepth(TreeNode* root) {
    int depth = 0;
    while (root->left) {
        root = root->left;
        depth++;
    }
    return depth;
}

bool exists(TreeNode* root, int depth, int idx) {
    int left = 0, right = pow(2, depth) - 1;
    for (int i = 0; i < depth; ++i) {
        int mid = left + (right - left) / 2;
        if (idx <= mid) {
            root = root->left;
            right = mid;
        } else {
            root = root->right;
            left = mid + 1;
        }
    }
    return root != NULL;
}
int countNodes(TreeNode* root) {
    if (!root) return 0;
    int depth = getDepth(root);
    if (depth == 0) return 1;

    int left = 1, right = pow(2, depth) - 1;
```

```
    while (left <= right) {
        int mid = left + (right - left) / 2;
        if (exists(root, depth, mid)) {
            left = mid + 1;
        } else {
            right = mid - 1;
        }
    }
    return pow(2, depth) - 1 + left;
}

TreeNode* insertLevelOrder(int arr[], int n, int i) {
    TreeNode* root = NULL;
    if (i < n && arr[i] != -1) {
        root = new TreeNode(arr[i]);
        root->left = insertLevelOrder(arr, n, 2 * i + 1);
        root->right = insertLevelOrder(arr, n, 2 * i + 2);
    }
    return root;
}

int main() {
    int n;
    cout << "Enter the number of elements in the array (including null as -1): ";
    cin >> n;

    int* arr = new int[n];
    cout << "Enter the elements of the array: ";
    for (int i = 0; i < n; i++) {
        cin >> arr[i];
    }

    TreeNode* root = insertLevelOrder(arr, n, 0);

    cout << "Number of nodes: " << countNodes(root) << endl;

    delete[] arr;
    return 0;
}
```

Output:

Enter the number of elements in the array (including null as -1): 6
Enter the elements of the array: 1 2 3 4 5 6
Number of nodes: 6

Code 3:

```cpp
#include <iostream>
#include <queue>
using namespace std;

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

int maxDepth(TreeNode* root) {
    if (root == NULL) return 0;
    int leftDepth = maxDepth(root->left);
    int rightDepth = maxDepth(root->right);
    return max(leftDepth, rightDepth) + 1;
}

TreeNode* insertLevelOrder(int arr[], int n, int i) {
    TreeNode* root = NULL;
    if (i < n && arr[i] != -1) {
        root = new TreeNode(arr[i]);
        root->left = insertLevelOrder(arr, n, 2 * i + 1);
        root->right = insertLevelOrder(arr, n, 2 * i + 2);
    }
    return root;
}

int main() {
    int n;
    cout << "Enter the number of elements in the array (including null as -1): ";
    cin >> n;

    int* arr = new int[n];
    cout << "Enter the elements of the array: ";
    for (int i = 0; i < n; i++) {
        cin >> arr[i];
    }

    TreeNode* root = insertLevelOrder(arr, n, 0);

    cout << "Maximum Depth: " << maxDepth(root) << endl;

    delete[] arr;
    return 0;
}
```

Output:

```
Enter the number of elements in the array (including null as -1): 7
Enter the elements of the array: 1 -1 2 3 6 7 2
Maximum Depth: 3
```

Code 4:

```cpp
#include <iostream>
#include <queue>
using namespace std;

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

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

TreeNode* insertLevelOrder(int arr[], int n, int i) {
    TreeNode* root = NULL;
    if (i < n && arr[i] != -1) {
        root = new TreeNode(arr[i]);
        root->left = insertLevelOrder(arr, n, 2 * i + 1);
        root->right = insertLevelOrder(arr, n, 2 * i + 2);
    }
    return root;
}

int main() {
    int n;
    cout << "Enter the number of elements in the array (including null as -1): ";
    cin >> n;

    int* arr = new int[n];
    cout << "Enter the elements of the array: ";
    for (int i = 0; i < n; i++) {
        cin >> arr[i];
    }

    TreeNode* root = insertLevelOrder(arr, n, 0);

    cout << "Preorder Traversal: ";
    preorderTraversal(root);
    cout << endl;

    delete[] arr;
```

```
    return 0;
}
```

Output:

Enter the number of elements in the array (including null as -1): 5
Enter the elements of the array: 2 1 3 6 7 8
Preorder Traversal: 2 1 6 7 3