CHANDIGARH COLLEGE OF ENGINEERING & TECHNOLOGY (DEGREE WING)

Government institute under Chandigarh (UT) Administration, affiliated to Punjab University, Chandigarh

Department of Computer Science & Engineering

Semester: CSE 3rd

SUBJECT: Data Structures Practical (CS351)

<u>Problem 7: Case Study of Binary Tree</u>

Submitted by:	**Submitted to:**
Bhavyam Dhand	Dr. R.B. Patel
(CO23316)	(Professor)
Date of Practical:2-Sep	**Date of Submission:**23-Sep

INDEX

CODE

```cpp
#include <iostream>
#include <string>
#include <fstream>
#include <cmath>
#include <ctime>
#include <bitset>
using namespace std;
// Function to write in Log file
void LogFile(const string& event) {
    ofstream BinFile("Machine_Code_Stack.txt", ios_base::app);
    if (BinFile.is_open()) {
        for (char c : event) {
            BinFile << bitset<8>(c) << " ";
        }
        BinFile << endl;
        BinFile.close();
    }
}
struct BTree{
    int Data;
    BTree* LST;
    BTree* RST;
};
void EnterNode(BTree*x, int d)
{
    x->Data = d;
    x->LST = NULL;
    x->RST = NULL;
}
BTree* CreateTree(BTree* head)
{
    int d;
    cout << "Enter Node Data: "; cin >> d;
    if (d == -1) // If user enters -1, it signifies no node should be added.
        return NULL;
    head = new BTree; // Allocate memory for a new node
    EnterNode(head, d);
    cout << "Enter the Left Node of current Node " << d << ": ";
    head->LST = CreateTree(head->LST);
    if (head->LST==NULL)
        return head;
    cout << "Enter the Right Node of current BTree " << d << ": ";
    head->RST = CreateTree(head->RST);
    return head;
}
// To print Binary Tree height
int TreeHeight(BTree* head)
{
    if (head==NULL)
        return 0;
    else {
        int LHeight = TreeHeight(head->LST);
        int RHeight = TreeHeight(head->RST);
        if (LHeight > RHeight)
```

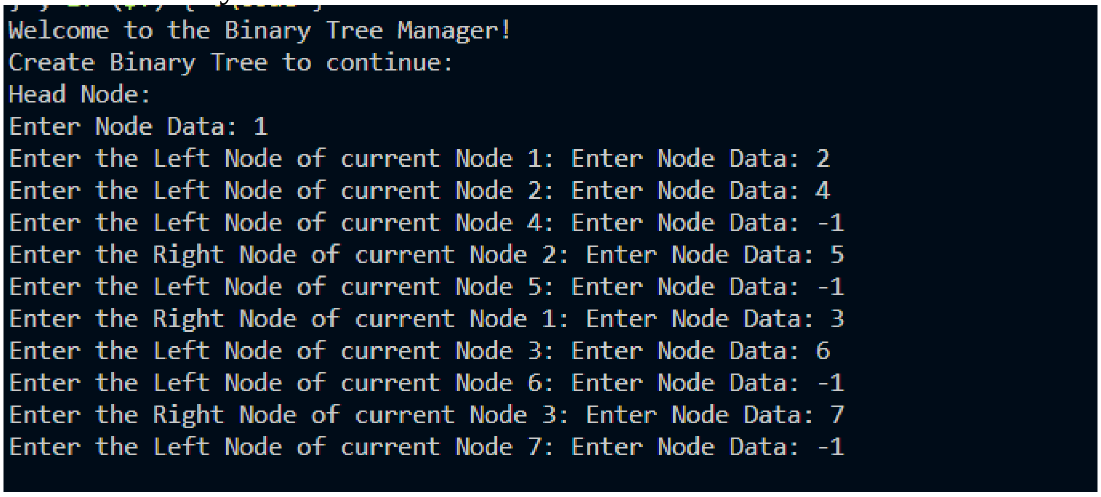
```
            return (LHeight + 1);
        else
            return (RHeight + 1);
    }
}
void PrintCurrentLevel(BTree* head, int level)
{
    if (head == NULL)
        return;
    if (level == 1)
        cout << head->Data << " ";
    else if (level > 1)
    {
        PrintCurrentLevel(head->LST, level - 1);
        PrintCurrentLevel(head->RST, level - 1);
    }
}
void PrintLevelOrder(BTree* head)
{
    int h = TreeHeight(head), i;
    for (i = 1; i <= h; i++) // Start at level 1
        PrintCurrentLevel(head, i);
}
// For Preorder Traversal (head->left->right)
void PreOrder(BTree* head)
{
    if (head==NULL)
        return;
    cout << head->Data << " ";
    PreOrder(head->LST);
    PreOrder(head->RST);
}
// For Inorder Traversal (left->head->right)
void InOrder(BTree* head)
{
    if (head==NULL)
        return;
    InOrder(head->LST);
    cout << head->Data << " ";
    InOrder(head->RST);
}
// For Postorder Traversal (left->right->head)
void PostOrder(BTree* head)
{
    if (head==NULL)
        return;
    PostOrder(head->LST);
    PostOrder(head->RST);
    cout << head->Data << " ";
}
void DeleteTree(BTree* head)
{
    if (head == NULL)
        return;

    // First delete both subtrees
```

```cpp
        DeleteTree(head->LST);
        DeleteTree(head->RST);
        // Then delete the node itself
        delete head;
    }
    int main()
    {
        LogFile("Starting Program");
        int z;
        BTree* Head = NULL;
        cout << "Welcome to the Binary Tree Manager!" << endl;
        cout << "Create Binary Tree to continue:\nHead Node:" << endl;
        Head = CreateTree(Head);
        LogFile("Created a Binary Tree");
        // Userbase
        while (true)
        {
            cout << "\nEnter your Commands!" << endl;
            cout << "1. Print Tree" << endl;
            cout << "2. Preorder Traversal" << endl;
            cout << "3. Inorder Traversal" << endl;
            cout << "4. Postorder Traversal" << endl;
            cout<<"Click Any Else Button to close Program!"<<endl;
            cin >> z;
            switch (z)
            {
            case 1:
                cout << "Printing Tree: ";
                PrintLevelOrder(Head);
                LogFile("Printing Binary Tree as is");
                break;
            case 2:
                cout << "Preorder Traversal is: ";
                PreOrder(Head);
                cout << endl;
                LogFile("Printing Binary Tree Node in Preorder traversal form");
                break;
            case 3:
                cout << "Inorder Traversal is: ";
                InOrder(Head);
                cout << endl;
                LogFile("Printing Binary Tree Node in Inorder traversal form");
                break;
            case 4:
                cout << "Postorder Traversal is: ";
                PostOrder(Head);
                cout << endl;
                LogFile("Printing Binary Tree Node in Postorder traversal form");
                break;
            default:
                LogFile("Closing Program");
                DeleteTree(Head);
                return 0;
            }
        }
    }
```

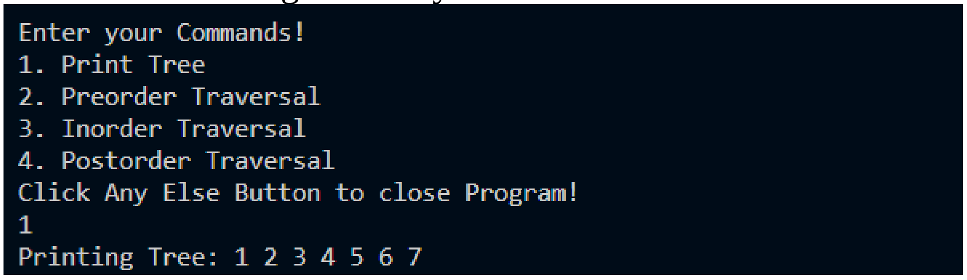
Output:

1. Create a Binary Tree:



```
Welcome to the Binary Tree Manager!
Create Binary Tree to continue:
Head Node:
Enter Node Data: 1
Enter the Left Node of current Node 1: Enter Node Data: 2
Enter the Left Node of current Node 2: Enter Node Data: 4
Enter the Left Node of current Node 4: Enter Node Data: -1
Enter the Right Node of current Node 2: Enter Node Data: 5
Enter the Left Node of current Node 5: Enter Node Data: -1
Enter the Right Node of current Node 1: Enter Node Data: 3
Enter the Left Node of current Node 3: Enter Node Data: 6
Enter the Left Node of current Node 6: Enter Node Data: -1
Enter the Right Node of current Node 3: Enter Node Data: 7
Enter the Left Node of current Node 7: Enter Node Data: -1
```

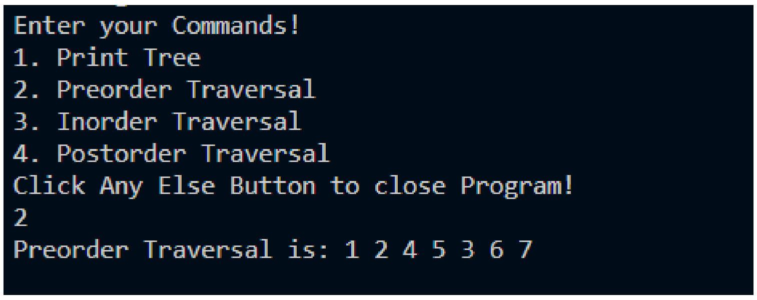
2. Level-wise Printing of Binary Tree:



```
Enter your Commands!
1. Print Tree
2. Preorder Traversal
3. Inorder Traversal
4. Postorder Traversal
Click Any Else Button to close Program!
1
Printing Tree: 1 2 3 4 5 6 7
```

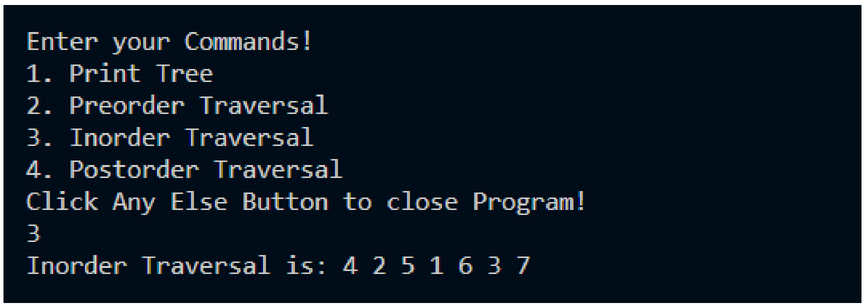
3. Traversal Algorithms

3.1. Preorder Traversal



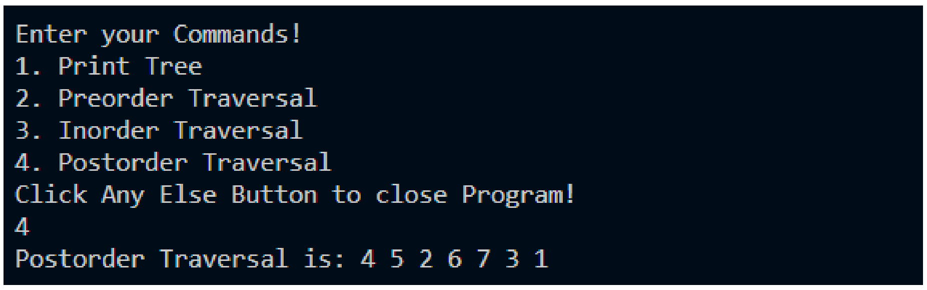
```
Enter your Commands!
1. Print Tree
2. Preorder Traversal
3. Inorder Traversal
4. Postorder Traversal
Click Any Else Button to close Program!
2
Preorder Traversal is: 1 2 4 5 3 6 7
```

3.2. Inorder Traversal



```
Enter your Commands!
1. Print Tree
2. Preorder Traversal
3. Inorder Traversal
4. Postorder Traversal
Click Any Else Button to close Program!
3
Inorder Traversal is: 4 2 5 1 6 3 7
```

3.3. Postorder Traversal



```
Enter your Commands!
1. Print Tree
2. Preorder Traversal
3. Inorder Traversal
4. Postorder Traversal
Click Any Else Button to close Program!
4
Postorder Traversal is: 4 5 2 6 7 3 1
```

Log File as Machine Code:

```
01010011 01110100 01100001 01110010 01110100 01101001 01101110 01100111
00100000 01010000 01110010 01101111 01100111 01110010 01100001 01101101
01000011 01110010 01100101 01100001 01110100 01100101 01100100 00100000
01100001 00100000 01000010 01101001 01101110 01100001 01110010 01111001
00100000 01010100 01110010 01100101 01100101
01010000 01110010 01101001 01101110 01110100 01101001 01101110 01100111
00100000 01000010 01101001 01101110 01100001 01110010 01111001 00100000
01010100 01110010 01100101 01100101 00100000 01100001 01110011 00100000
01101001 01110011
01010000 01110010 01101001 01101110 01110100 01101001 01101110 01100111
00100000 01000010 01101001 01101110 01100001 01110010 01111001 00100000
01010100 01110010 01100101 01100101 00100000 01001110 01101111 01100100
01100101 00100000 01101001 01101110 00100000 01010000 01110010 01100101
01101111 01110010 01100100 01100101 01110010 00100000 01110100 01110010
01100001 01110110 01100101 01110010 01110011 01100001 01101100 00100000
01100110 01101111 01110010 01101101
01010000 01110010 01101001 01101110 01110100 01101001 01101110 01100111
00100000 01000010 01101001 01101110 01100001 01110010 01111001 00100000
01010100 01110010 01100101 01100101 00100000 01001110 01101111 01100100
01100101 00100000 01101001 01101110 00100000 01001001 01101110 01101111
01110010 01100100 01100101 01110010 00100000 01110100 01110010 01100001
01110110 01100101 01110010 01110011 01100001 01101100 00100000 01100110
01101111 01110010 01101101
01010000 01110010 01101001 01101110 01110100 01101001 01101110 01100111
00100000 01000010 01101001 01101110 01100001 01110010 01111001 00100000
01010100 01110010 01100101 01100101 00100000 01001110 01101111 01100100
01100101 00100000 01101001 01101110 00100000 01010000 01101111 01110011
01110100 01101111 01110010 01100100 01100101 01110010 00100000 01110100
01110010 01100001 01110110 01100101 01110010 01110011 01100001 01101100
00100000 01100110 01101111 01110010 01101101
01000011 01101100 01101111 01110011 01101001 01101110 01100111 00100000
01010000 01110010 01101111 01100111 01110010 01100001 01101101
01010011 01110100 01100001 01110010 01110100 01101001 01101110 01100111
00100000 01010000 01110010 01101111 01100111 01110010 01100001 01101101
01010011 01110100 01100001 01110010 01110100 01101001 01101110 01100111
00100000 01010000 01110010 01101111 01100111 01110010 01100001 01101101
01010011 01110100 01100001 01110010 01110100 01101001 01101110 01100111
00100000 01010000 01110010 01101111 01100111 01110010 01100001 01101101
01000011 01110010 01100101 01100001 01110100 01100101 01100100 00100000
01100001 00100000 01000010 01101001 01101110 01100001 01110010 01111001
00100000 01010100 01110010 01100101 01100101
01010000 01110010 01101001 01101110 01110100 01101001 01101110 01100111
00100000 01000010 01101001 01101110 01100001 01110010 01111001 00100000
01010100 01110010 01100101 01100101 00100000 01100001 01110011 00100000
01101001 01110011
01010011 01110100 01100001 01110010 01110100 01101001 01101110 01100111
00100000 01010000 01110010 01101111 01100111 01110010 01100001 01101101
01010011 01110100 01100001 01110010 01110100 01101001 01101110 01100111
00100000 01010000 01110010 01101111 01100111 01110010 01100001 01101101
01010011 01110100 01100001 01110010 01110100 01101001 01101110 01100111
00100000 01010000 01110010 01101111 01100111 01110010 01100001 01101101
01000011 01110010 01100101 01100001 01110100 01100101 01100100 00100000
01100001 00100000 01000010 01101001 01101110 01100001 01110010 01111001
00100000 01010100 01110010 01100101 01100101
01010000 01110010 01101111 01100111 01110010 01100001 01101101
```