Experiment no: 7 Date:

Aim: Write a program to create a binary tree and traverse it in in order, pre order and post order

Programming Language: C/C++/Java

Theory: A Tree is a non-linear data structure and a hierarchy consisting of a collection of nodes such that each node of the tree stores a value and a list of references to other nodes called as its children.

Creating tree structure:

Each node in a linked list consists of three fields, INFO, LEFT, and RIGHT pointer

- LEFT [k] contains the location of the left child of node N.
- INFO [k] contains the data at the node N.
- RIGHT [k] contains the location of right child of node N.

Tree traversal: Traversing a tree means visiting and outputting the value of each node in a particular order. The major importance of tree traversal is that there are multiple ways of carrying out traversal operations unlike linear data structures like arrays, bitmaps, matrices where traversal is done in a linear order. Each of these methods of traversing a tree have a particular order they follow:

- For Inorder, traverse from the left subtree to the root then to the right subtree.
- For Preorder, traverse from the root to the left subtree then to the right subtree.
- For Post order,traverse from the left subtree to the right subtree then to the root.

Code :

```
#include <iostream>

using namespace std;

struct node {

  int data;

  node *left;

  node *right;

};

struct node *create(int c) {

  struct node *newnode = new node;
```

```
  newnode->data = c;
  newnode->left = NULL;
  newnode->right = NULL;
  cout << "Node created : " << c << endl;
  return newnode;
}
void insertleft(struct node *root, int c)  { root->left = create(c); }
void insertright(struct node *root, int c)  { root->right = create(c); }
void preorder(struct node *root) {
  if (root == NULL) {
    return;
  }
  cout << " " << root->data;
  preorder(root->left);
  preorder(root->right);
}
void inorder(struct node *root) {
  if (root == NULL) {
    return;
  }
  inorder(root->left);
  cout << " " << root->data;
  inorder(root->right);
}
```

```
void postorder(struct node *root) {
  if (root == NULL) {
    return;
  }
  postorder(root->left);
  postorder(root->right);
  cout << " " << root->data;
}
int main() {
  int n;
  cout<<"Jay And Janay"<<endl;
  cout << "Enter the root node" << endl;
  cin >> n;
  struct node *root = create(n);
  insertleft(root, 15);
  insertright(root, 20);
  insertleft(root->left, 17);
  insertleft(root->right, 11);
  insertright(root->right, 16);
  insertright(root->right->right, 19);
  insertleft(root->left->left, 11);
  insertright(root->left->left, 14);

  cout<<"Preorder list :  ";
  preorder(root);
```

```
 cout << endl;
 cout<<"Inorder list :  ";
 inorder(root);
 cout << endl;
 cout<<"Postorder list :  ";
 postorder(root);
 return 0;
}
```

```
Jay And Janay
Enter the root node
5
Node created : 5
Node created : 15
Node created : 20
Node created : 17
Node created : 11
Node created : 16
Node created : 19
Node created : 11
Node created : 14
Preorder list :   5 15 17 11 14 20 11 16 19
Inorder list :   11 17 14 15 5 11 20 16 19
Postorder list :   11 14 17 15 11 19 16 20 5
```

Result and Conclusion: