```
#include <bits/stdc++.h>

using namespace std;

class treeNode
{

public :

   int data;
   treeNode * leftchild;
   treeNode * rightchild;

   treeNode(int value)
   {

      data = value;
      leftchild = NULL;
      rightchild = NULL;
   }



};

void spacePrint ( int level){
   for ( int i = 0; i<level; i++){
      cout<<"   ";
   }
}

void printTree(treeNode * root, int level){
   if (root == NULL)
   {
      return;
   }
   if ( root->leftchild == NULL && root->rightchild == NULL){
      cout<<root->data<<endl;
   }
   else
   {
   cout<<endl;
   spacePrint(level);
```

```
    cout<< " Root: "<< root->data<<endl;
    }

    if ( root->leftchild != NULL)
    {
    spacePrint(level);
    cout<< " Left: ";
    printTree(root->leftchild, level+1);
    }

    if (root->rightchild != NULL)
    {
    spacePrint(level);
    cout<< " Right: ";
    printTree(root->rightchild, level+1);
    }
}

void inOrder(treeNode * root, string &chk){
    if (root == NULL)  return;

      inOrder(root->leftchild,chk);
      chk+=to_string(root->data);
      inOrder(root->rightchild,chk);
}

void PreOrder(treeNode * root, string &chk){
    if (root == NULL)  return;

      chk+=to_string(root->data);
      PreOrder(root->leftchild,chk);
      PreOrder(root->rightchild,chk);
}

void PostOrder(treeNode * root, string &chk){
    if (root == NULL)  return;

      PostOrder(root->leftchild,chk);
      PostOrder(root->rightchild,chk);
      chk+=to_string(root->data);
}
```

```
int main() {

    int n;
    cin>>n;

    treeNode * allNodes[n];

    for ( int i =0; i<n; i++){
        allNodes[i] = new treeNode(-1);
    }

    for ( int i =0; i<n; i++)
    {

        int value,left,right;
        cin>>value>>left>>right;
        allNodes[i]->data = value;

        if(left>n-1 || right>n-1)
        {
            cout<< " Invalid Index "<<endl;
            break;
        }
        if (left != -1){
            allNodes[i]->leftchild=allNodes[left];
        }

         if (right != -1){
            allNodes[i]->rightchild=allNodes[right];
        }
    }

    printTree(allNodes[0], 0);
    string inordertravarsal ="";
    string preordertravarsal ="";
    string postordertravarsal ="";

    inOrder(allNodes[0],inordertravarsal);
    PreOrder(allNodes[0],preordertravarsal);
    PostOrder(allNodes[0],postordertravarsal);


    cout<< " Inorder Travarsal "<<inordertravarsal<<endl;
    cout<< " Preorder Travarsal "<<preordertravarsal<<endl;
```

```
    cout<< " Postorder Travarsal "<<postordertravarsal<<endl;

    return 0;
}
```