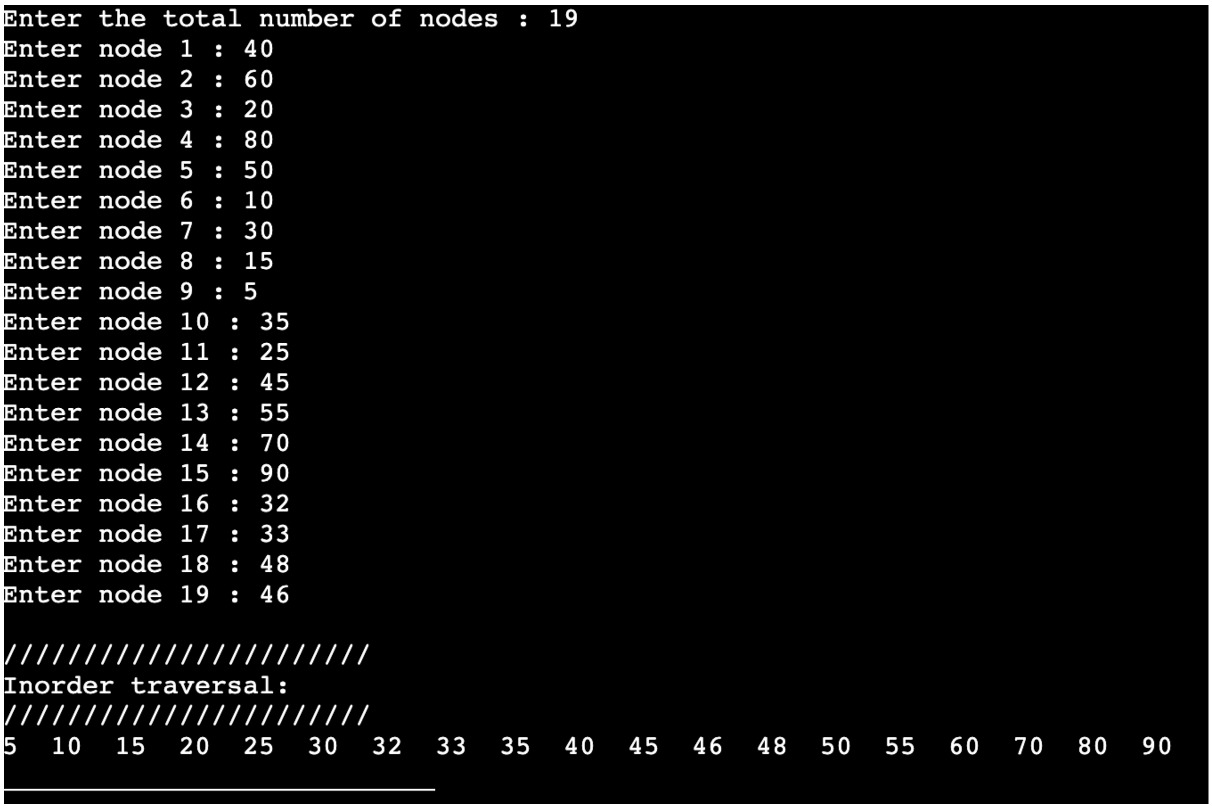
Assignment - 2
BST Insertion and Deletion

CODE:

```
#include <iostream>

using namespace std;

struct BST_node{
   int data;
   BST_node* Lch;
   BST_node* Rch;
};

BST_node* new_node(int number){
   BST_node* node = new BST_node();
   node -> data = number;
   node -> Lch = NULL;
   node -> Rch = NULL;
   return node;
}

BST_node* inst(BST_node* root, int number){
   if (root == NULL){
      root = new_node(number);
   }
   else if (number <= root -> data){
      root -> Lch = inst(root -> Lch, number);
   }
   else{
      root -> Rch = inst(root -> Rch, number);
   }
   return root;
}

void display_inorder(BST_node* root){

   if (root==NULL)
   {
      return;
   }
   display_inorder(root->Lch);
   cout<<root->data<<"  ";
   display_inorder(root->Rch);
}

BST_node* Successor(BST_node* root){
    if (root->Lch != NULL){
        root = Successor(root->Lch);
    }
    return root;
}

/*BST_node* Predecessor(BST_node* root){
     if (root->Rch != NULL){
         root = Predecessor(root->Rch);
     }
     return root;
}*/

BST_node* del(BST_node* root, int number){
    if (root == NULL){
        return root;
    }

    else if(number < root->data){
        root -> Lch = del(root -> Lch, number);
    }
    else if(number > root->data){
        root -> Rch = del(root -> Rch, number);
    }
    else{
        if((root->Lch == NULL) && (root->Rch == NULL)){
            delete root;
            root = NULL;
            return root;
        }
        else if(root->Lch == NULL){
            BST_node* tmp_node = root;
            root = root->Rch;
            delete tmp_node;
            return root;
        }
        else if(root->Rch == NULL){
            BST_node* tmp_node = root;
            root = root->Lch;
            delete tmp_node;
            return root;
        }
        else{
            BST_node* tmp_node = Successor(root->Rch);
            //BST_node* tmp_node = Predecessor(root->Lch);
            root->data = tmp_node->data;
            root->Rch = del(root->Rch, tmp_node->data);
        }
    }
    return root;

}

int main()
{
    BST_node* root = NULL;
    int node_number, element;
    cout<<"Enter the total number of nodes : ";
    cin>>node_number;
    for(int i=0; i<node_number ; i++){
        cout<<"Enter node "<<i+1<< " : ";
        cin>>element;
        root = inst(root,element);
    }

    cout<<endl<<"///////////////////////"<<endl;
    cout<<"Inorder traversal:"<<endl;
    cout<<"///////////////////////"<<endl;
    display_inorder(root);
    cout<<endl<<"________________________________________"<<endl<<endl;
    cout<<"Enter the node to delete : ";
    cin>>element;
    root = del(root,element);
    cout<<endl<<"///////////////////////"<<endl;
    cout<<"Inorder traversal:"<<endl;
    cout<<"///////////////////////"<<endl;
    display_inorder(root);
    cout<<endl<<"________________________________________"<<endl<<endl;
    cout<<"Enter the node to delete : ";
    cin>>element;
    root = del(root,element);
    cout<<endl<<"///////////////////////"<<endl;
    cout<<"Inorder traversal:"<<endl;
    cout<<"///////////////////////"<<endl;
    display_inorder(root);
    return 0;
}
```

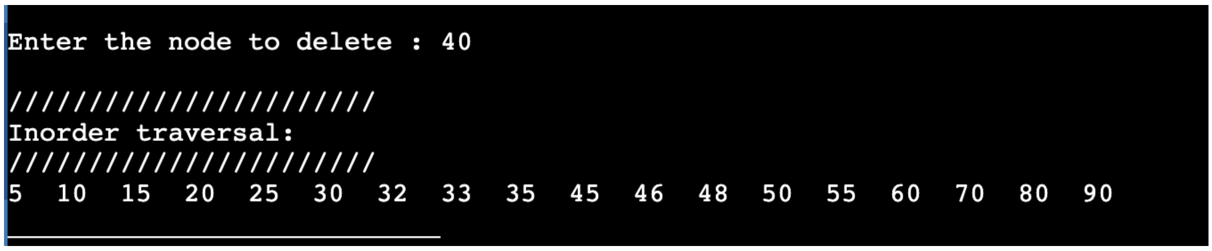
OUTPUT:

Insertion and Display in Inorder:



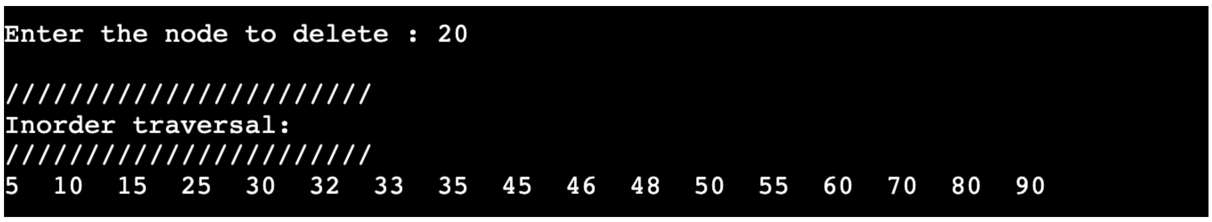
```
Enter the total number of nodes : 19
Enter node 1 : 40
Enter node 2 : 60
Enter node 3 : 20
Enter node 4 : 80
Enter node 5 : 50
Enter node 6 : 10
Enter node 7 : 30
Enter node 8 : 15
Enter node 9 : 5
Enter node 10 : 35
Enter node 11 : 25
Enter node 12 : 45
Enter node 13 : 55
Enter node 14 : 70
Enter node 15 : 90
Enter node 16 : 32
Enter node 17 : 33
Enter node 18 : 48
Enter node 19 : 46

///////////////////////
Inorder traversal:
///////////////////////
5  10  15  20  25  30  32  33  35  40  45  46  48  50  55  60  70  80  90
___________________________
```

Delete 40:



```
Enter the node to delete : 40

///////////////////////
Inorder traversal:
///////////////////////
5  10  15  20  25  30  32  33  35  45  46  48  50  55  60  70  80  90
___________________________
```

Delete 20:



```
Enter the node to delete : 20

///////////////////////
Inorder traversal:
///////////////////////
5  10  15  25  30  32  33  35  45  46  48  50  55  60  70  80  90
```