CS 5343 Algorithm Analysis and Data Structures

Assignment #2

Submitted by

Naga Mutya Kumar Kumtsam

(nxk210028)

Binary Tree Insertion and Deletion of nodes

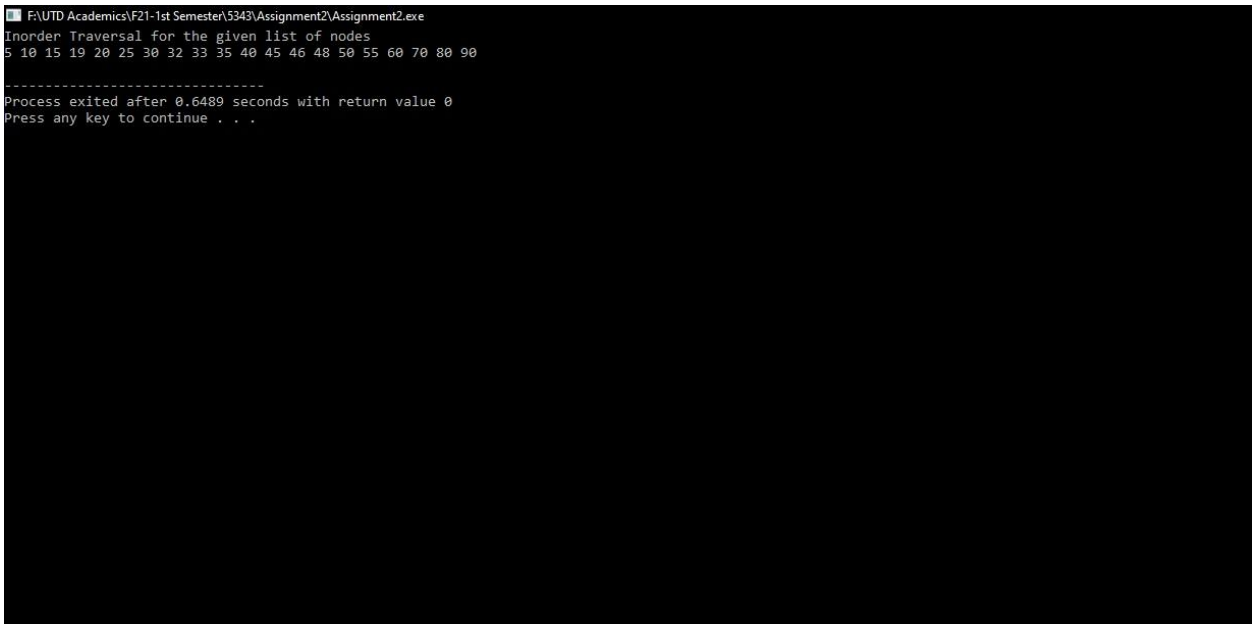
Program :

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

struct Node
{
	int head;
	struct Node *left, *right;
};
struct Node * createnode(int head)
{
	struct Node * val = new Node;
	val->head = head ;
	val->left = NULL ;
	val->right = NULL ;
	return val ;
}
struct Node * insertnode(struct Node* Node,int head)
{
	if (Node == NULL)
	{
		return createnode(head);
	}
	if (head < Node->head)
	{
		Node->left = insertnode(Node->left,head);
	}
	else
	{
		Node->right = insertnode(Node->right,head);
	}
	 return Node ;
}
void InorderTraversal(Node* value)
{
	if (value != NULL)
	{
		InorderTraversal(value->left);
		cout << (value->head) << " " ;
		InorderTraversal(value->right);
	}
}
```

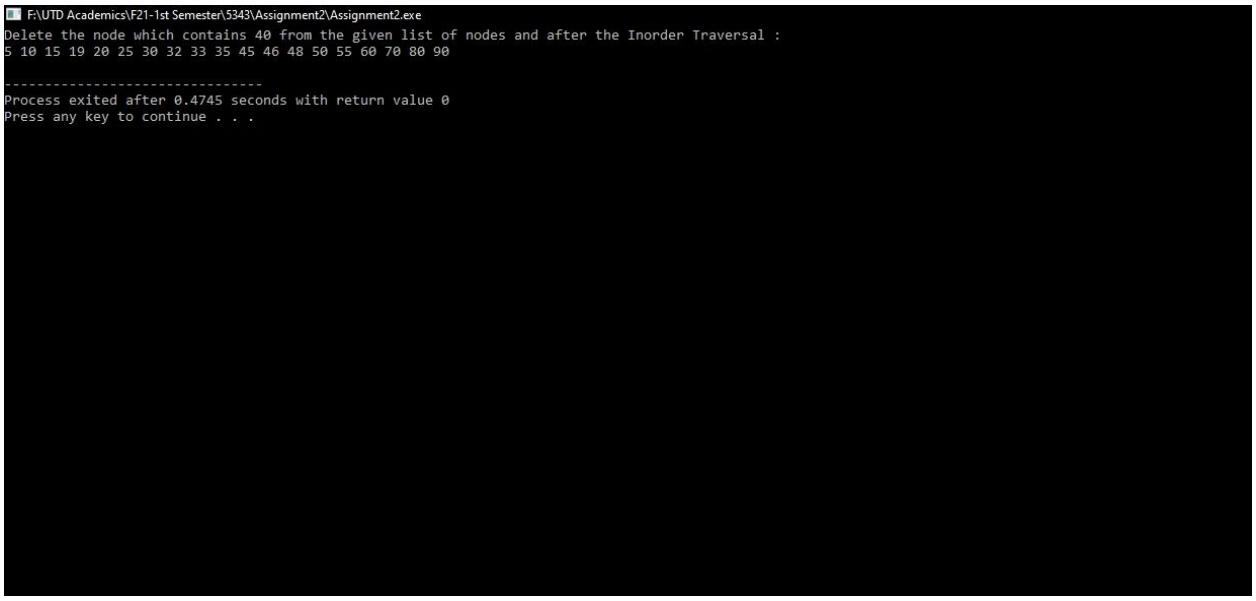
```
struct Node* deletenode(struct Node * node,int value)
{
	if(node == NULL)
	{
		return node;
	}

	if (value<node->head)
	{
		node->left = deletenode(node->left,value);
	}
	else if (value>node->head)
	{
		node->right = deletenode(node->right,value);
	}

	if(node->right == NULL and node->left == NULL)
	{
		return NULL ;
	}
	else if (node->right == NULL)
	{
		Node * temp = node->left;
		delete node;
		return temp;
	}
	else if (node->left == NULL)
	{
		Node * temp = node->right;
		delete node;
		return temp;
	}
	else
	{
		Node* parent = node;
		Node* pre = node->left;
		while (pre->right != NULL)
		{
			parent = pre;
			pre = pre->right;
		}

		if (parent != node)
			parent->right = pre->left;
```

```
                else
                        parent->left = pre->left;

                node->head = pre->head;
                delete pre;
                return node;
        }
}
int main()
{
        struct Node * head = NULL;
        int arr[] = {40, 60, 20, 80, 50, 10, 30, 15, 5, 35, 25, 45, 55, 70, 90, 32, 33, 48, 46};
        int n,dnode ;
        n = sizeof(arr)/sizeof(arr[0]);
        for (int i=0 ; i<= n; i++)
        {
                head = insertnode(head,arr[i]);
        }

                cout<<"Inorder Traversal for the given list of nodes\n";
                InorderTraversal(head);
                cout << endl ;

        cout << "Delete the node which contains 40 from the given list of nodes and after the Inorder
Traversal :\n";
        head = deletenode(head,40);
        InorderTraversal(head);
                cout << endl;

        cout << "Delete the node which contains 20 from the given list of nodes and after the Inorder
Traversal : \n";
        head = deletenode(head,20);
                InorderTraversal(head);
}
```

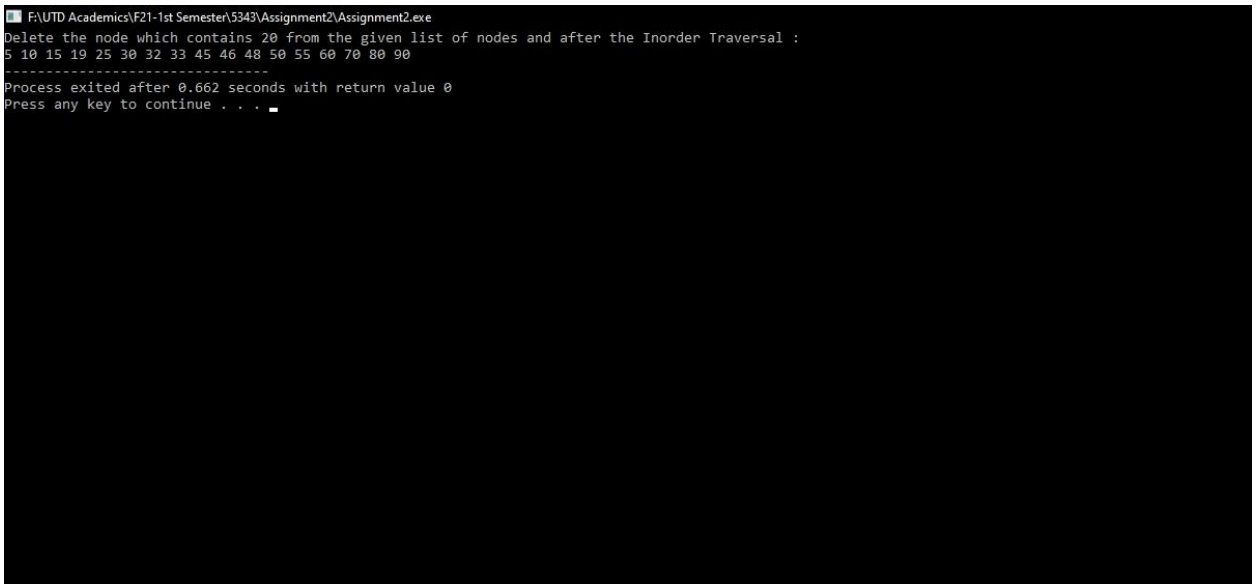
Inorder Traversal for the given list of Nodes



Inorder Traversal after the deletion of 40 from the list of nodes.



Inorder Traversal after the deletion of 20 from the list of nodes.



All executions