ASSIGNMENT NO.2

Aim:

Construct a threaded binary search tree by inserting values in the given order and traverse it in inorder traversal using threads.

Objective:

To understand the following Concepts of Threaded Binary Search Tree (TBT):

i. Creating a TBT using tree data structure.
ii. Inorder traversal using threads.

THEORY:

Threaded Binary Tree:

Inorder traversal of a Binary tree can either be done using recursion or with the use of a auxiliary stack. The idea of threaded binary trees is to make inorder traversal faster and do it without stack and without recursion. A binary tree is made threaded by making all right child pointers that would normally be NULL point to the inorder successor of the node (if it exists).

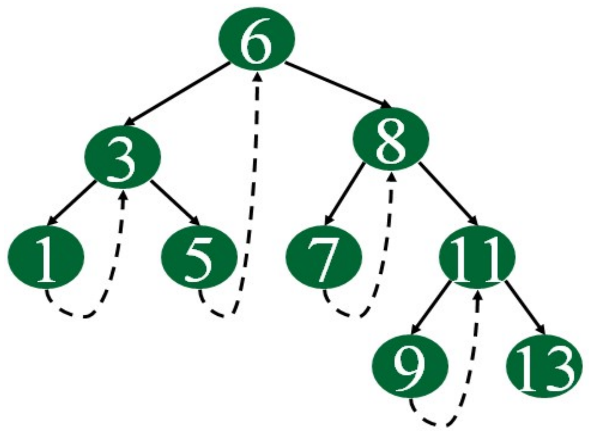
There are two types of threaded binary trees.

Single Threaded: Where a NULL right pointers is made to point to the inorder successor (if successor exists)

Double Threaded: Where both left and right NULL pointers are made to point to inorder predecessor and inorder successor respectively. The predecessor threads are useful for reverse inorder traversal and postorder traversal.

The threads are also useful for fast accessing ancestors of a node.

Following diagram shows an example Single Threaded Binary Tree. The dotted lines represent threads.



ALGORITHM:

Non recursive Inorder traversal for a Threaded Binary Tree

```
1. curr-node node  leftmost (root)
 2. While (curr_node != Null)
        a. print (curr_node)
        b. If (curr_node.RTag == 0) then
                    curr_node <- curr_node.right
                    go to step 2.
        c. else     curr_node <- leftmost(curr_node.right)
                    go to step 2.
```

CODE:

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

```
class treenode
{
friend class tbt;
int data;
treenode *lptr,*rptr;
int lbit,rbit;
public:
}*dummy;
class tbt
{
treenode *root;
public:
   treenode *create()
{
   treenode *a=new treenode;
   a->lptr=a->rptr=NULL;
   a->lbit=a->rbit=1;
   cout<<"\n Enter the data";
   cin>>a->data;
   return a;

}
   tbt()
   {
      root=NULL;
```

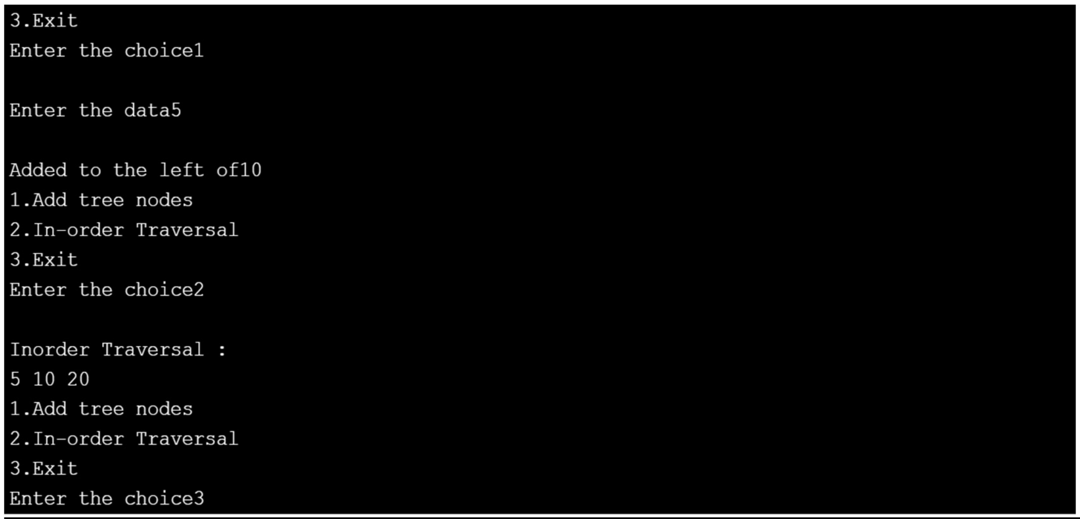
```
}
void accept()
{
    treenode *a=new treenode;
    a=create();
    if(root==NULL)
    {
        /*dummy->data=-9999;
        dummy->lbit=0;
        dummy->rbit=0;
        dummy->lptr=a;
        dummy->rptr=dummy;
        */
        root=a;
    }
    else
    {
        treenode *p=root;

        while(1)
        {
            if((a->data)<(p->data))
              {
                if(p->lbit==1)
                   break;
```

```
         else
             p=p->lptr;
        }
        else
        {
            if(p->rbit==1)
             break;
         else
             p=p->rptr;
        }
    }
    if((a->data)<(p->data))
      {                cout<<"\n Added to the left of"<<(p->data);
        a->lptr=p->lptr;
        a->rptr=p;
        p->lptr=a;
        p->lbit=0;
      }
      else
       {
            cout<<"\n Added to the  right of"<<(p->data);
             a->rptr=p->rptr;
             a->lptr=p;
             p->rbit=0;
             p->rptr=a;
            }
        }
    }
treenode* leftmost(treenode *a)
{
while(a->lbit==0)
{
a=a->lptr;
}
return a;
}
void  traversal()
{
   cout<<"\n Inorder Traversal : \n";
treenode *a=root;
a=leftmost(root);
while(a!=NULL)
{
cout<<" "<<a->data;
if(a->rbit==1)
{
a=(a->rptr);
}
else
{
```

```
a=leftmost(a->rptr);
}
}
}
};
int main()
{
   tbt s;
   int c;
   do
   {
   cout<<"\n 1.Add tree nodes \n 2.In-order Traversal \n 3.Exit";
   cout<<"\n Enter the choice";
   cin>>c;
   switch(c)
   {
      case 1:s.accept();break;
      case 2:s.traversal();break;
      case 3:break;
   }
   }while(c!=3);
}
```

OUTPUT:

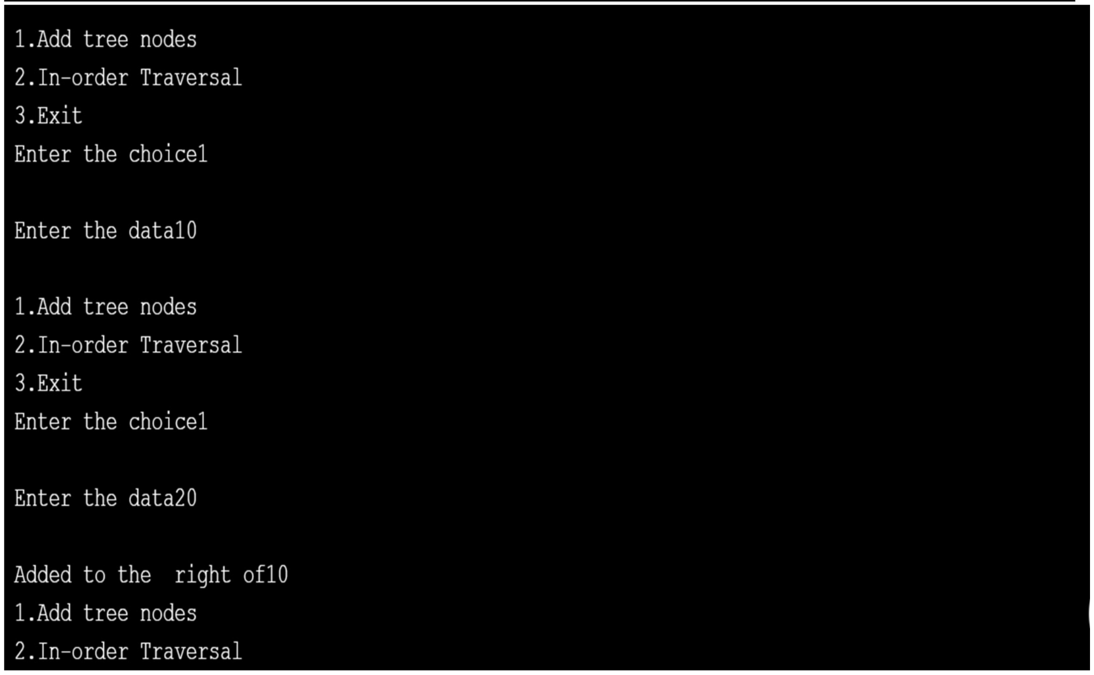


```
3.Exit
Enter the choice1

Enter the data5

Added to the left of10
1.Add tree nodes
2.In-order Traversal
3.Exit
Enter the choice2

Inorder Traversal :
5 10 20
1.Add tree nodes
2.In-order Traversal
3.Exit
Enter the choice3
```



```
1.Add tree nodes
2.In-order Traversal
3.Exit
Enter the choice1

Enter the data10

1.Add tree nodes
2.In-order Traversal
3.Exit
Enter the choice1

Enter the data20

Added to the  right of10
1.Add tree nodes
2.In-order Traversal
```

CONCLUSION:

Study and operations like creating TBT and Inorder traversal using threads was performed successfully.