

## ASSIGNMENT 2

### AIM:-

Construct a threaded binary tree by inserting value in a given order and traverse it in in order traversal using threads.

### OBJECTIVE:-

To implement the concept of threaded binary tree and perform in order traversal.

### 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).

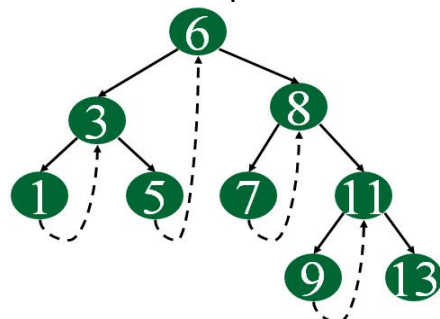
There are two types of threaded binary trees.

**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.



#### C representation of a Threaded Node

Following is C representation of a single threaded node.

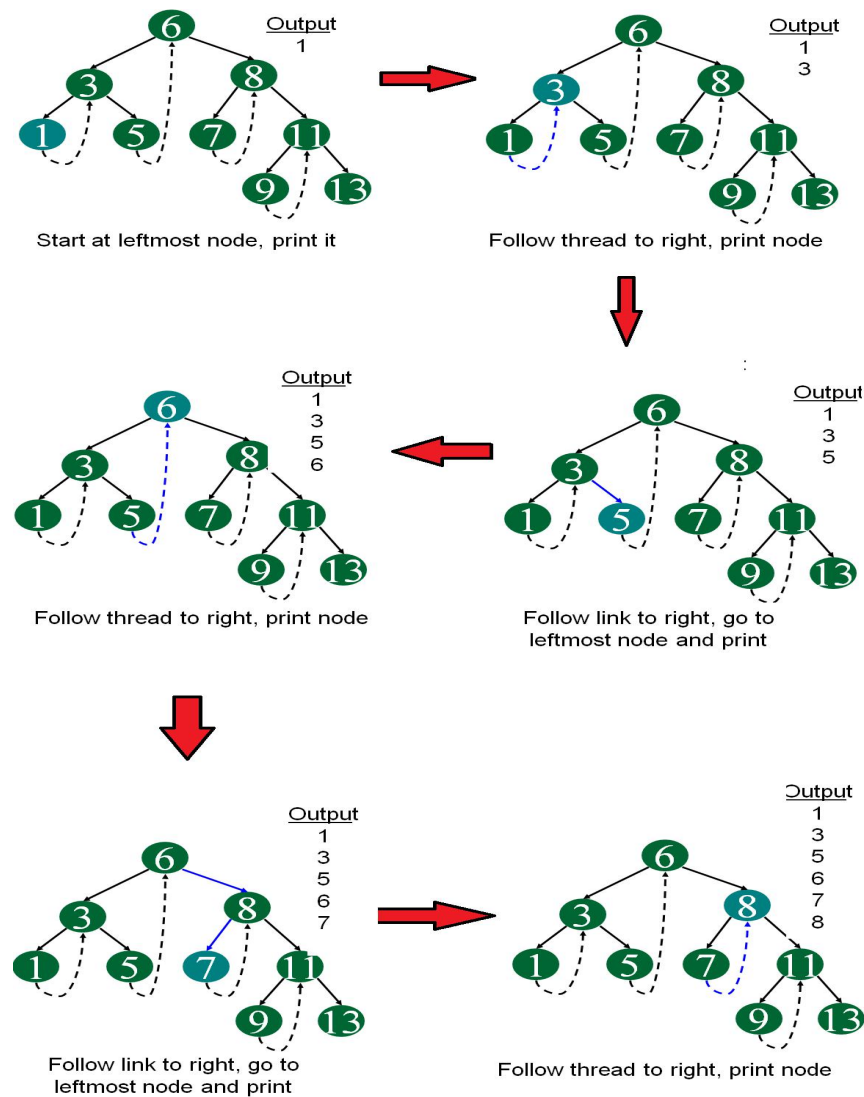
```
struct Node
```

```
{
    int data;
    Node *left, *right;
    bool rightThread;
}
```

Since right pointer is used for two purposes, the boolean variable rightThread is used to indicate whether right pointer points to right child or inorder successor. Similarly, we can add leftThread for a double threaded binary tree.

### IN ORDER TRAVERSAL:-

Following diagram demonstrates inorder order traversal using threads.



**continue same way for remaining node.....**

**ALGORITHM:-****1.INSERT A NODE**

```

Void create()
    headnode = new treeNode();
    headnode->lthread = false;
    headnode->rthread = true;
    headnode->left = headnode->right = headnode;
}
void insert(int data)
{
    treeNode* nn = new treeNode;
    nn->left = NULL;
    nn->right = NULL;
    nn->data = data;
    if(headnode == headnode->left && headnode ==
headnode->right)
    {
        nn->left = headnode;
        headnode->left = nn;
        nn->lthread = headnode->lthread;
        headnode->lthread = true;
        nn->right = headnode;
    }

    else
    {
        treeNode* current = headnode->left;
        while(true)
        {
            if(current->data > nn->data)
            {
                if(current->lthread == false)
                {
                    nn->left = current->left;
                    current->left = nn;
                    nn->lthread = current->lthread;
                    current->lthread = true;
                    nn->right = current;
                    break;
                }
            }
        }
    }
}

```

```

        }

        else
            current = current->left;
    }

    else
    {
        if(current->rthread == false)
        {
            nn->right = current->right;
            current->right = nn;
            nn->rthread = current->rthread;
            current->rthread = true;
            nn->left = current;
            break;
        }
        else
        {
            current = current->right;
        }
    }
}

}

}

```

## 2. NON RECURSIVE INORDER TRAVERSAL

```

void inorder()
{
    treeNode* current = headnode->left;
    while (current->lthread == true)
        current = current->left;

    while (current != headnode)
    {
        cout << current->data;
        if(current->rthread == false)

```

```

        {
            current = current->right;
        }
        else
        {
            current = current->right;
            while (current->lthread != false)
                current = current->left;
        }
    }
}

```

### 3.INORDER RECURSIVE TRAVERSAL

```

void reclnorder(treeNode* node)
{
    if (node != headnode )
    {
        reclnorder(node->left);
        cout << node->data << " ";
        reclnorder(node->right);
    }
}

```

#### CODE:-

```

#include<iostream>
using namespace std;

```

```

struct treeNode
{
    int data;
    treeNode* left;
    treeNode* right;
    bool lthread;
    bool rthread;
};

```

```

class Tree
{
    private :

```

```

        treeNode* headnode = NULL;
public :
    Tree()
    {
        headnode = new treeNode();
        headnode->lthread = false;
        headnode->rthread = true;
        headnode->left = headnode->right = headnode;
    }
    void insert(int data)
    {
        treeNode* nn = new treeNode;
        nn->left = NULL;
        nn->right = NULL;
        nn->data = data;
        if(headnode == headnode->left && headnode ==
headnode->right)
        {
            nn->left = headnode;
            headnode->left = nn;
            nn->lthread = headnode->lthread;
            headnode->lthread = true;
            nn->right = headnode;
        }

        else
        {
            treeNode* current = headnode->left;
            while(true)
            {
                if(current->data > nn->data)
                {
                    if(current->lthread == false)
                    {
                        nn->left = current->left;
                        current->left = nn;
                        nn->lthread = current->lthread;
                        current->lthread = true;
                        nn->right = current;
                        break;

```

```

        }

        else
            current = current->left;
    }

    else
    {
        if(current->rthread == false)
        {
            nn->right = current->right;
            current->right = nn;
            nn->rthread = current->rthread;
            current->rthread = true;
            nn->left = current;
            break;
        }
        else
        {
            current = current->right;
        }
    }
}

}

}

}

void inorder()
{
    treeNode* current = headnode->left;
    while (current->lthread == true)
        current = current->left;

    while (current != headnode)
    {
        cout << current->data;
        if(current->rthread == false)
        {

```

```

        current = current->right;
    }
    else
    {
        current = current->right;
        while (current->lthread != false)
            current = current->left;
    }

}
}
void reInorder(treeNode* node)
{
    if (node != headnode )
    {
        reInorder(node->left);
        cout << node->data << " ";
        reInorder(node->right);
    }

}
treeNode* getRoot()
{
    return headnode->left;
}
}tree;

int main()
{
    char choice='y';
    int ch;
    int data;

    while(choice == 'y')
    {
        cout << "\tMENU" << endl;
        cout << "\t1.insert Binay Search tree node" << endl;
        cout << "\t2.Inorder traversal" << endl;
    }
}

```



```

        cout << "\tenter your choice" << endl;
        cin >> ch;
        switch(ch)
        {
            case 1:
                cout << "Enter node data: " << endl;
                cin >> data;
                tree.insert(data);
                break;
            case 2 :
                tree.inorder();
                break;
            default :
                cout << "\tINVALID CHOICE" << endl;

        }
        cout << "\tDo you wish to continue" << endl;
        cout << "\tIf yes enter y" << endl;
        cin >> choice;
    }
}

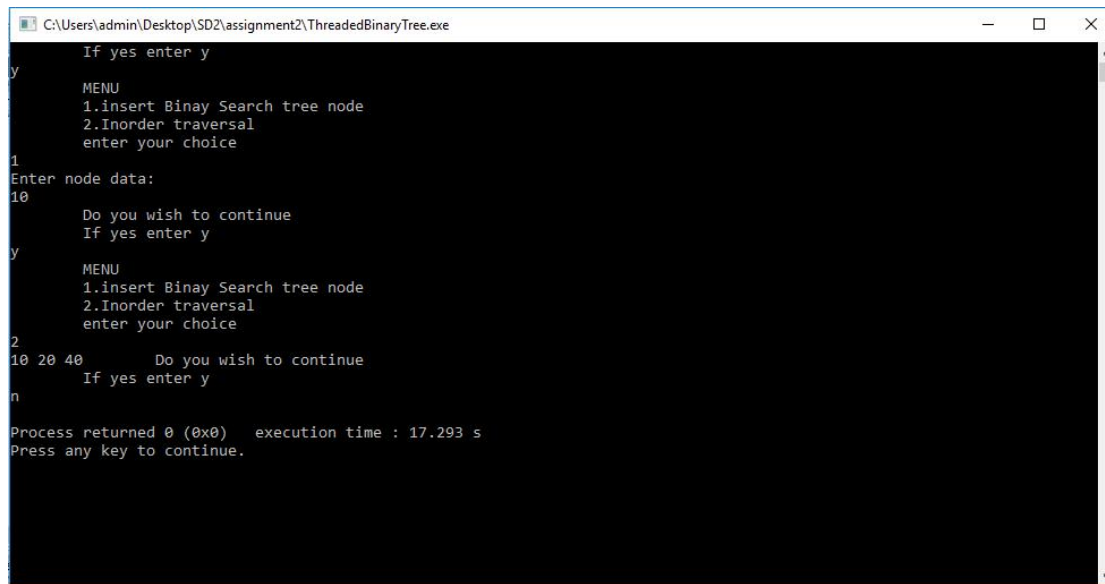
```

### OUTPUT:-

```

C:\Users\admin\Desktop\SD2\assignment2\ThreadedBinaryTree.exe
MENU
1.insert Binay Search tree node
2.Inorder traversal
enter your choice
1
Enter node data:
20
Do you wish to continue
If yes enter y
y
MENU
1.insert Binay Search tree node
2.Inorder traversal
enter your choice
1
Enter node data:
40
Do you wish to continue
If yes enter y
y
MENU
1.insert Binay Search tree node
2.Inorder traversal
enter your choice
1
Enter node data:
10
Do you wish to continue
If yes enter y
y

```



```
C:\Users\admin\Desktop\SD2\assignment2\ThreadedBinaryTree.exe
y
If yes enter y
MENU
1.insert Binay Search tree node
2.Inorder traversal
enter your choice
1
Enter node data:
10
Do you wish to continue
If yes enter y
y
MENU
1.insert Binay Search tree node
2.Inorder traversal
enter your choice
2
10 20 40 Do you wish to continue
If yes enter y
n
Process returned 0 (0x0) execution time : 17.293 s
Press any key to continue.
```

### CONCLUSION:-

We have successfully implemented threaded binary tree and functions like insertion, traversal.