

ASSIGNMENT 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 study the concept of threaded binary tree.

Theory:-

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).

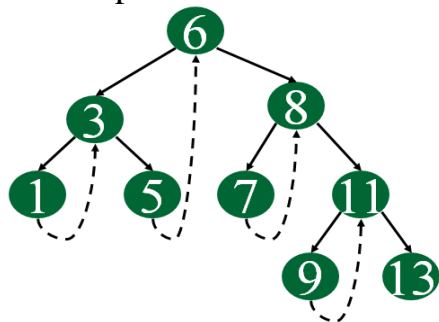
There are two types of threaded binary trees.

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:-

Let **tmp** be the newly inserted node. There can be three cases during insertion:

Case 1: Insertion in empty tree

Both left and right pointers of tmp will be set to NULL and new node becomes the root.

```
root = tmp;
```

```
tmp -> left = NULL;
```

```
tmp -> right = NULL;
```

Case 2: When new node inserted as the left child

After inserting the node at its proper place we have to make its left and right threads points to inorder predecessor and successor respectively. The node which was inorder successor. So the left and right threads of the new node will be-

```
tmp -> left = par -> left;
```

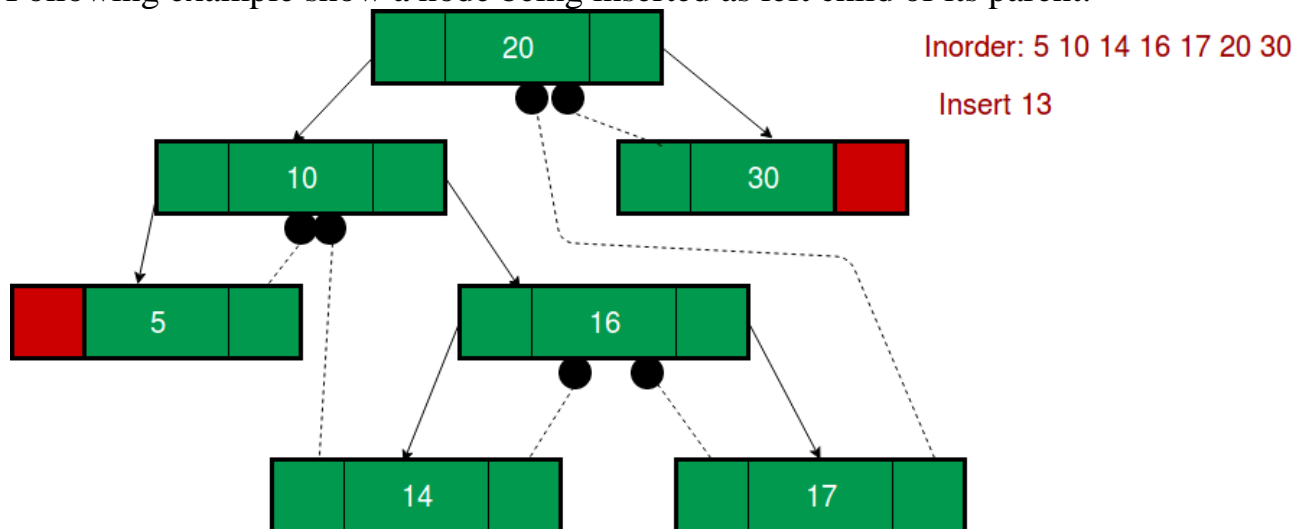
```
tmp -> right = par;
```

Before insertion, the left pointer of parent was a thread, but after insertion it will be a link pointing to the new node.

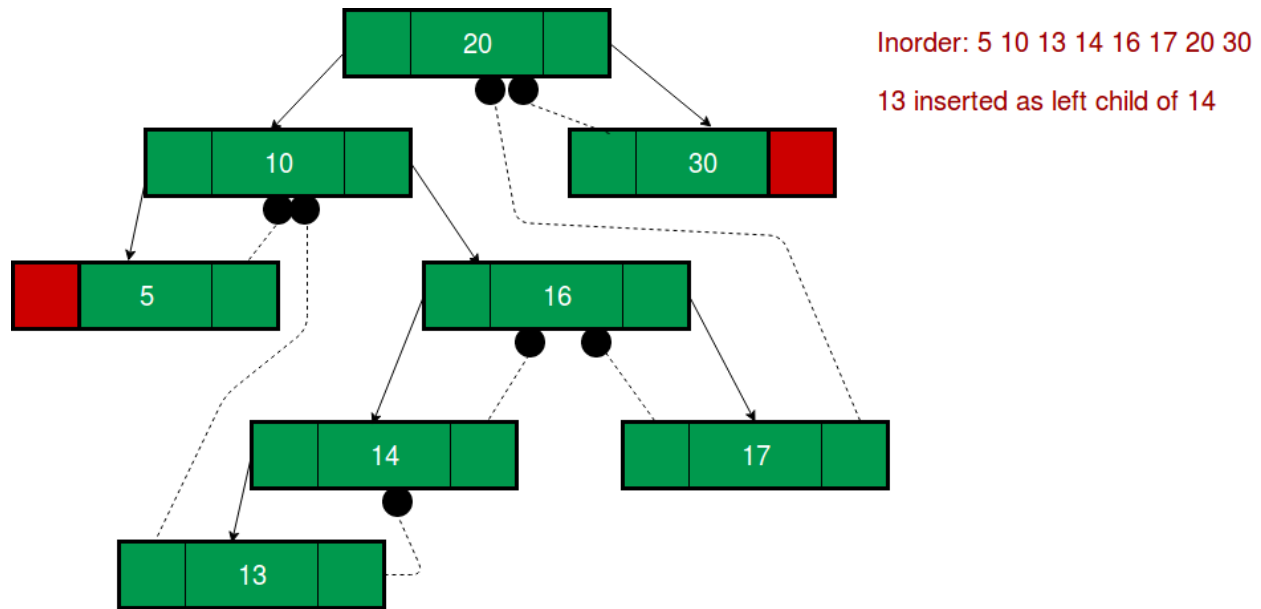
```
par -> lthread = false;
```

```
par -> left = temp;
```

Following example show a node being inserted as left child of its parent.



After insertion of 13,



Predecessor of 14 becomes the predecessor of 13, so left thread of 13 points to 10.
Successor of 13 is 14, so right thread of 13 points to left child which is 13.
Left pointer of 14 is not a thread now, it points to left child which is 13.

Case 3: When new node is inserted as the right child

The parent of tmp is its inorder predecessor. The node which was inorder successor of the parent is now the inorder successor of this node tmp. So the left and right threads of the new node will be-

```
tmp -> left = par;
```

```
tmp -> right = par -> right;
```

Before insertion, the right pointer of parent was a thread, but after insertion it will be a link pointing to the new node.

```
par -> rthread = false;
```

```
par -> right = tmp;
```

Program Code:-

```
#include<iostream>
```

```
using namespace std;
```

```
class ttree
```

```
{
```

```
    private:
```

```
        struct thtree
```

```
        {
```

```
            int left;
```

```
            thtree *leftchild;
```

```
            int data;
```

```
            thtree *rightchild;
```

```
            int right;
```

```
        }*th_head;
```

```
    public:
```

```
        ttree();
```

```
        void insert(int num);
```

```
        void inorder();
```

```
};
```

```
ttree::ttree()
```

```
{
```

```
    th_head=NULL;
```

```
}
```

```
void ttree::insert(int num)
```

```
{
```

```
    thtree *head=th_head,*p,*z;
```

```
    z=new thtree;
```

```
    z->left=true;
```

```
    z->data=num;
```

```
    z->right=true;
```

```
    if(th_head==NULL)
```

```
    {
```

```
        head=new thtree;
```

```
        head->left=false;
```

```
        head->leftchild=z;
        head->data=-9999;
        head->rightchild=head;
        head->right=false;

        th_head=head;
        z->leftchild=head;
        z->rightchild=head;
    }
    else
    {
        p=head->leftchild;
        while(p!=head)
        {
            if(p->data > num)
            {
                if(p->left!=true)
                p=p->leftchild;
                else
                {
                    z->leftchild=p->leftchild;
                    p->leftchild=z;

                    p->left=false;
                    z->right=true;
                    z->rightchild=p;
                    return;
                }
            }
            else
            {
                if(p->data < num)
                {
                    if(p->right!=true)
                    p=p->rightchild;
                    else
                    {
                        z->rightchild=p->rightchild;
                        p->rightchild=z;
```

```
p->right=false;
z->left=true;
z->leftchild=p;
return;
}
}
}
}
}
void ttree::inorder()
{
    thtree *a;
    a=th_head->leftchild;
    while(a!=th_head)
    {
        while(a->left==false)

            a=a->leftchild;
        cout<<a->data<<"\t";

        while(a->right==true)
        {
            a=a->rightchild;

            if(a==th_head)
                break;

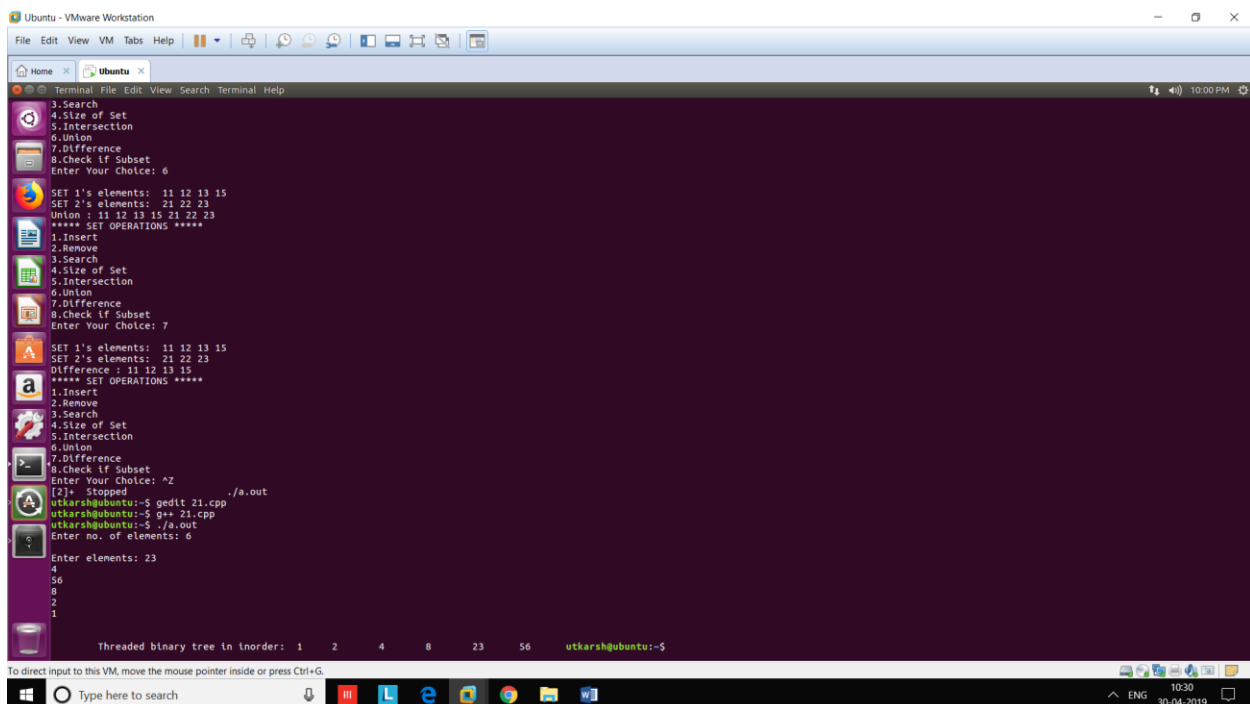
            cout<<a->data<<"\t";
        }
        a=a->rightchild;
    }
}

int main()
{
    ttree th;
```

```
int n,e;
cout<<"Enter no. of elements: ";
cin>>n;
cout<<"\nEnter elements: ";
for(int i=0;i<n;i++)
{
    cin>>e;
    th.insert(e);
}

cout<<"\n\n\tThreaded binary tree in inorder: ";
th.inorder();
}
```

Output Screenshots:-



The screenshot shows a terminal window titled 'Ubuntu - VMware Workstation'. The terminal displays the following output:

```
3.Search
4.Size of Set
5.Intersection
6.Union
7.Difference
8.Check If Subset
Enter Your Choice: 6
SET 1's elements: 11 12 13 15
SET 2's elements: 21 22 23
Union : 11 12 13 15 21 22 23
***** SET OPERATIONS *****
1.Insert
2.Remove
3.Search
4.Size of Set
5.Intersection
6.Union
7.Difference
8.Check If Subset
Enter Your Choice: 7
SET 1's elements: 11 12 13 15
SET 2's elements: 21 22 23
Difference : 11 12 13 15
***** SET OPERATIONS *****
1.Insert
2.Remove
3.Search
4.Size of Set
5.Intersection
6.Union
7.Difference
8.Check If Subset
Enter Your Choice: ^Z
[2]+ Stopped ./a.out
utkarsh@ubuntu:~$ gedit 21.cpp
utkarsh@ubuntu:~$ g++ 21.cpp
utkarsh@ubuntu:~$ ./a.out
Enter no. of elements: 6
Enter elements: 23
56
8
2
1
Threaded binary tree in inorder: 1 2 4 8 23 56
utkarsh@ubuntu:~$
```

Conclusion:- Thus,we have studied Threaded binary tree.