

# SKILL DEVELOPMENT LAB-II(2018-19)

## 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 :

We have to implement threaded binary search tree using BST data structure.

**Theory :** 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).

### Applications :

Threaded binary can be used where stack space is limited. Threaded binary is used where fastest traversal is the main requirement .

### Program :

```
#include<iostream>

using namespace std;

class TBT

{

int data;

int lth,rth;

TBT *lptr,*rptr;

public:

void Create(int);

void Insert(TBT*,TBT*);
```

## SKILL DEVELOPMENT LAB-II(2018-19)

```
void Display_inorder(TBT*);
```

```
}*root=NULL,*headnode;
```

```
void TBT::Create(int y)
```

```
{
```

```
TBT *nn=new TBT;
```

```
nn->data=y;
```

```
nn->lptr=nn->rptr=NULL;
```

```
nn->lth=nn->rth=1; //1=thread
```

```
if(root==NULL)
```

```
{
```

```
root=nn;
```

```
headnode=new TBT;
```

```
headnode->data=0;
```

```
headnode->lptr=root;
```

```
headnode->rptr=headnode;
```

```
headnode->lth=headnode->rth=1;
```

```
root->lptr=root->rptr=headnode;
```

```
}
```

```
else
```

```
Insert(root,nn);
```

```
}
```

```
void TBT::Insert(TBT* temp, TBT* nn)
```

```
{
```

```
if(nn->data<temp->data)
```

## SKILL DEVELOPMENT LAB-II(2018-19)

```
{
if(temp->lth==1)
{
nn->lptr=temp->lptr;    //nn->lptr pointing to headnode
temp->lptr=nn;
nn->rptr=temp;
temp->lth=0;
}
else Insert(temp->lptr,nn);
}
else if(nn->data>temp->data)
{
if(temp->rth==1)
{
nn->rptr=temp->rptr;    //nn->rptr pointing to headnode
temp->rptr=nn;
nn->lptr=temp;
temp->rth=0;
}
else Insert(temp->rptr,nn);
}}

void TBT::Display_inorder(TBT* head)
{
TBT *current;
current=head->lptr;
```

## SKILL DEVELOPMENT LAB-II(2018-19)

```
while(current->lth!=1)

    current=current->lptr;

while(current!=head)

{

    cout<<current->data<<" ";

    if(current->rth==1)

        current=current->rptr;

    else

    {

        current=current->rptr;

        while(current->lth==0)

            current=current->lptr;

    }

}

}

int main()

{

    TBT t;

    int i,z,y;

    cout<<endl<<"How many numbers you want to enter : ";cin>>z;

    cout<<"\n enter numbers : ";

    i=0;

    while(i<z)

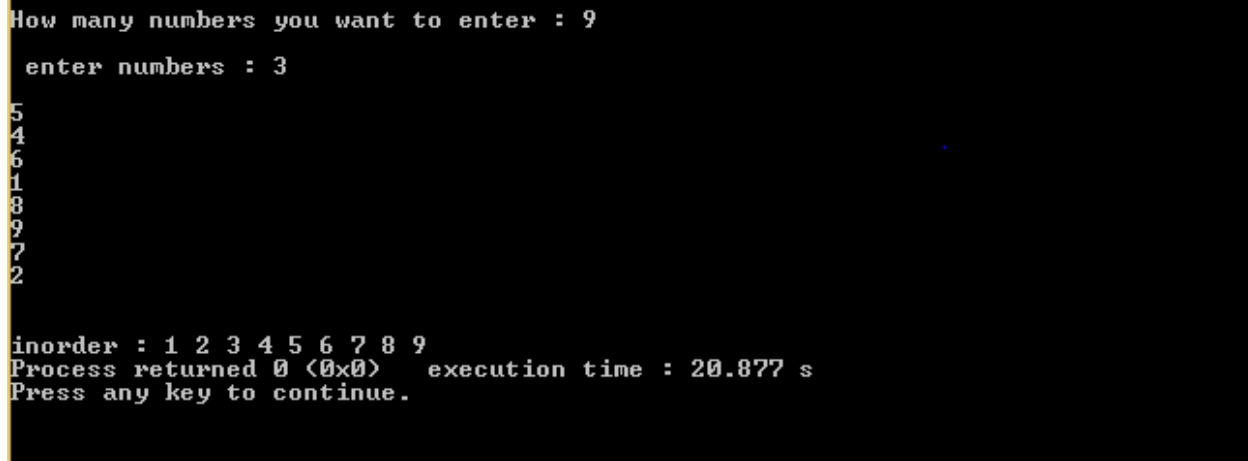
    {

        cin>>y;
```

## SKILL DEVELOPMENT LAB-II(2018-19)

```
t.Create(y);  
  
i++;  
  
}  
  
cout<<endl<<"\ninorder : ";  
  
t.Display_inorder(headnode);  
  
return 0;  
  
}
```

### Output :



```
How many numbers you want to enter : 9  
enter numbers : 3  
5  
4  
6  
1  
8  
9  
7  
2  
  
inorder : 1 2 3 4 5 6 7 8 9  
Process returned 0 (0x0)   execution time : 20.877 s  
Press any key to continue.
```

**Conclusion :** Thus we have studied threaded binary search tree using BST data structure.