Name : Onkar Shinde
Batch : C2
Roll No : COSC26

Assignment 5

Input :

```
#include <iostream>
using namespace std;
class TBT; class
node
{ node *left,*right; int
      data; bool
      rbit,lbit; public:

node()
      { left=NULL;
            right=NULL;
            rbit=lbit=0;
      }

      node(int d)
{

            left=NULL;
          right=NULL;
      rbit=lbit=0;
      data=d;
 }

      friend class TBT;
 };

class TBT
{ node *root; //acts as a dummy node
      public:
      TBT() //dummy node initialization
{ root=new node(9999); root-
            >left=root; root-
            >rbit=1; root-
            >lbit=0; root-
            >right=root;
      }

      void create(); void
      insert(int data); node
      *inorder_suc(node *); void
      inorder_traversal();

};

void TBT::inorder_traversal()
{ node *c=root->left; while(c-
            >lbit==1)      c=c-
            >left;
            while(c!=root)
```

```
            { cout<<" "<<c->data;
                  c=inorder_suc(c);
            }
      }

node* TBT::inorder_suc(node *c)
      { if(c->rbit==0)
      return c->right;
      else c=c->right;
      while(c-
      >lbit==1)
      { c=c->left;
      } return
      c;
}


//-------- Create Method
void TBT::create()
{ int n;
      cout<<"\nEnter number of nodes:";
      cin>>n; for(int i=0;i<n;i++)
      { int info; cout<<"\nEnter
            data: ";
      cin>>info;
this->insert(info); }
}

void TBT::insert(int data)
{

 if(root->left==root&&root->right==root) //no node in tree
 {  node *p=new node(data);
      p->left=root->left;
       p->lbit=root->lbit; //0
p->rbit=0;          p->right=root-
>right;       root->left=p;      root-
>lbit=1;      cout<<"\nInserted
start"<<data;       return;

 }

       node *cur=new node;      cur=root-
>left;// root node        while(1)
       {

            if(cur->data<data)    //insert right
            {
                  node *p=new node(data);
                  if(cur->rbit==0)
                  {
      p->right=cur->right;    p->rbit=cur-
>rbit;
      p->lbit=0;        p->left=cur;
cur->rbit=1;        cur->right=p;
                        cout<<"\nInserted right "<<data;
                        return;
                  }
                  else
                        cur=cur->right;
            }
```

```
            if(cur->data>data) //insert left
            {
                  node *p=new node(data);
                  if(cur->lbit==0)
                  {
       p->left=cur->left;      p->lbit=cur-
>lbit;
                        p->rbit=0;
                        p->right=cur; //successor
       cur->lbit=1;      cur->left=p;
                        cout<<"\nInserted left"<<data;
                        return;
                  }
                  else
                        cur=cur->left;
            }
       }
}

int main() {  TBT
      t1;  int
      value;  int
      choice;  do
{   cout<<"\n1.Create Tree\n2.Inorder\n3.Exit\nEnter your choice: ";
      cin>>choice;   switch(choice)
        {   case 1:    t1.create();    break;
      case 2:    cout<<"\nInoder Traversal of
      TBT\n";    t1.inorder_traversal();
      break;   case 3:    break;   default:
      cout<<"\nWrong choice";
        }

 }    while(choice!=3)
return 0;
}
```

Output :

```
1.Create Tree
2.Inorder
3.Exit

Enter your choice: 1

Enter number of nodes:5

Enter data: 20

Inserted start20
Enter data: 10

Inserted left10
Enter data: 25

Inserted right 25
Enter data: 40

Inserted right 40
Enter data: 65
```

```
Inserted right 65
1.Create Tree
2.Inorder
3.Exit

Enter your choice: 2

Inoder Traversal of TBT

10 20 25 40 65

1.Create Tree
2.Inorder
3.Exit

Enter your choice: 3
```