**Name: Om Chandrakant Bhavsar**

**Class: SE-A**

**Roll No: COSA75**

**Practical No. 4**

---

```
#include<iostream>
#include<math.h>
using namespace std;

struct Bstnode
{
 int data;
 Bstnode *left = NULL;
 Bstnode *right = NULL;

};

class Btree
{

  int n;
  int x;
  int flag;

public:
  Bstnode * root;
 Btree()
 {
  root = NULL;
 }

 Bstnode *GetNewNode(int in_data)
 {
  Bstnode * ptr = new Bstnode();
  ptr->data = in_data;
  ptr->left = NULL;
  ptr->right = NULL;
  return ptr;
 }

 Bstnode *insert( Bstnode *temp , int in_data)
 {
  if( temp == NULL )
  {
```

```
 temp = GetNewNode(in_data);
 }
 else if( temp->data > in_data)
 {
  temp->left = insert(temp->left , in_data);
 }
 else
 {
  temp->right = insert( temp->right , in_data);
 }
 return temp;
}

void input()
{
 cout<<"ENTER NUMBER OF ELEMENTS IN THE BST : ";
 cin>>n;
 for(int i = 0 ; i < n ; i++)
 {
  cout<<"NUMBER = ";
  cin>>x;
  root = insert(root , x);
 }
}

int search(Bstnode *temp ,int in_data)
{
 if( temp != NULL)
 {
  if(temp->data == in_data)
  {
   cout<<":-- RECORD FOUND --:"<<endl;
   return 1;
  }
  else if(in_data < temp->data)
  {
   this->search(temp->left, in_data);
  }
  else if(in_data > temp->data)
  {
   this->search(temp->left , in_data);
  }
 }
 else
 {
   return 0;
```

```
 }
}


void minvalue(Bstnode *temp)
{
 while(temp->left != NULL)
 {
  temp = temp->left;
 }
 cout<<"MINIMUM VALUE = "<<temp->data<<endl;
}

void mirror(Bstnode *temp)
{
 if(temp == NULL)
 {
  return;
 }
 else
 {
  Bstnode *ptr;
  mirror(temp->left);
  mirror(temp->right);
  ptr = temp->left;
  temp->left = temp->right;
  temp->right = ptr;
 }
}

void display()
{
 cout<<endl<<"--- INORDER TRAVERSAL ---"<<endl;
 inorder(root);
 cout<<endl;
 cout<<endl<<"--- POSTORDER TRAVERSAL ---"<<endl;
 postorder(root);
 cout<<endl;
 cout<<endl<<"--- PREORDER TRAVERSAL ---"<<endl;
 preorder(root);
 cout<<endl;

}

void inorder(Bstnode *temp)
{
```

```
 if(temp != NULL)
 {
  inorder(temp->left);
  cout<<temp->data<<"  ";
  inorder(temp->right);
 }
}

void postorder(Bstnode *temp)
{
 if(temp != NULL)
 {
  postorder(temp->left);
  postorder(temp->right);
  cout<<temp->data<<" ";
 }
}

void preorder(Bstnode *temp)
{
 if(temp != NULL)
 {
  cout<<temp->data<<" ";
  preorder(temp->left);
  preorder(temp->right);
 }
}

int depth(Bstnode *temp)
{
 if(temp == NULL)
  return 0;
 return (max((depth(temp->left)),(depth(temp->right))) +1);
}
};

int main()
{
 Btree obj;
 obj.input();
 obj.display();
 int a = 0;
 a = obj.search(obj.root,10);
 if( a == 0)
 {
  cout<<"ELEMENT NOT FOUND"<<endl;
```

```
 }
 else
  cout<<"ELEMENT FOUND"<<endl;
 cout<<endl<<a<<endl;
 obj.minvalue(obj.root);
 obj.mirror(obj.root);
 obj.inorder(obj.root);
 //int d ;
 cout<<endl<<obj.depth(obj.root);
 //cout<<endl<<d<<endl;
 return 0;
}
```

Output:

```
ENTER NUMBER OF ELEMENTS IN THE BST : 5
NUMBER = 4
NUMBER = 2
NUMBER = 1
NUMBER = 7
NUMBER = 5

--- INORDER TRAVERSAL ---
1  2  4  5  7

--- POSTORDER TRAVERSAL ---
1 2 5 7 4

--- PREORDER TRAVERSAL ---
4 2 1 7 5
ELEMENT NOT FOUND

0
MINIMUM VALUE = 1
7  5  4  2  1
3
```