```
#include<iostream>
#include<stdlib.h>

using namespace std;

struct treeNode
{
   int data;
   treeNode *left;
   treeNode *right;
};
treeNode* FindMin(treeNode *node)
{
   if(node==NULL)
   {

      return NULL;
   }
   if(node->left)
      return FindMin(node->left);
   else
      return node;
}
treeNode* FindMax(treeNode *node)
{
   if(node==NULL)
   {

      return NULL;
   }
   if(node->right)
      return(FindMax(node->right));
   else
      return node;
}
treeNode *Insert(treeNode *node,int data)
{
   if(node==NULL)
   {
      treeNode *temp;
      temp=new treeNode;

      temp -> data = data;
      temp -> left = temp -> right = NULL;
      return temp;
   }
   if(data >(node->data))
   {
      node->right = Insert(node->right,data);
   }
```

```
    else if(data < (node->data))
    {
        node->left = Insert(node->left,data);
    }

    return node;
}
treeNode * Delet(treeNode *node, int data)
{
    treeNode *temp;
    if(node==NULL)
    {
        cout<<"Element Not Found";
    }
    else if(data < node->data)
    {
        node->left = Delet(node->left, data);
    }
    else if(data > node->data)
    {
        node->right = Delet(node->right, data);
    }
    else
    {

        if(node->right && node->left)
        {

            temp = FindMin(node->right);
            node -> data = temp->data;

            node -> right = Delet(node->right,temp->data);
        }
        else
        {

            temp = node;
            if(node->left == NULL)
                node = node->right;
            else if(node->right == NULL)
                node = node->left;
            free(temp);
        }
    }
    return node;
}
treeNode * Find(treeNode *node, int data)
{
    if(node==NULL)
    {
```

```
        return NULL;
    }
    if(data > node->data)
    {

        return Find(node->right,data);
    }
    else if(data < node->data)
    {

        return Find(node->left,data);
    }
    else
    {

        return node;
    }
}
void Inorder(treeNode *node)
{
    if(node==NULL)
    {
        return;
    }
    Inorder(node->left);
    cout<<node->data<<" ";
    Inorder(node->right);
}

int main()
{
    treeNode *root = NULL,*temp;
    int ch;

    while(1)
    {
        cout<<"\n1.Insert\n2.Delete\n3.Inorder\n4.Search\n5.Exit\n";
        cout<<"Enter ur choice:";
        cin>>ch;
        switch(ch)
        {
        case 1:
            cout<<"\nEnter element to be insert:";
            cin>>ch;
            root = Insert(root, ch);
            cout<<"\nElements in BST are:";
            Inorder(root);
            break;
        case 2:
```

```
            cout<<"\nEnter element to be deleted:";
            cin>>ch;
            root = Delet(root,ch);
            cout<<"\nAfter deletion elements in BST are:";
            Inorder(root);
            break;
        case 3:
            cout<<"\nInorder Travesals is:";
            Inorder(root);
            break;

        case 4:
            cout<<"\nEnter element to be searched:";
            cin>>ch;
            temp = Find(root,ch);
            if(temp==NULL)
            {
                cout<<"Element is not foundn";
            }
            else
            {
                cout<<"Element "<<temp->data<<" is Found\n";
            }
            break;
        case 5:
            exit(0);
            break;
        default:
            cout<<"\nEnter correct choice:";
            break;
        }
    }
    return 0;
}
```

```
Enter the number of nodes to be insert: 6

Please enter the numbers to be insert: 2 6 9 1 5 8

Binary Search Tree nodes in Inorder Traversal:   1  2  5  6  8

Process returned 0 (0x0)   execution time : 15.837 s
Press any key to continue.
```