Assignment 11 94COA124

1] What is BST?
→ BST is also called as binary search tree are also called as ordered or sorted binary tree, is a rooted binary tree data structure where internal nodes each store a key greater than all keys in the node left subtree and less than tree in it's right subtree.

2] Operations on BST?
→ Basic operation ~~on~~ BST;
a] ~~create~~ Insert :- create an empty node and insert a data in it.

b] create :- create an empty tree.

c] search :- It searches an element from the tree.
d] Delete :- It deletes an element (node) from the tree.
e] inorder :- It displays all the element in ascending order.
f] preorder :- It displays the data first and then traverse in the tree.
g] postorder :- ~~In d~~ It displays the data last and then traverse the tree first.

94COA124

Algorithm for searching :-
step 1:- Take the input from user.
step 2:- Traverse the tree and compare the data of every node with input data.
step 3:- If record is found display the msg record found and break the ~~execu~~ execution of search function.
step 4:- If record is not found display the msg record not found.

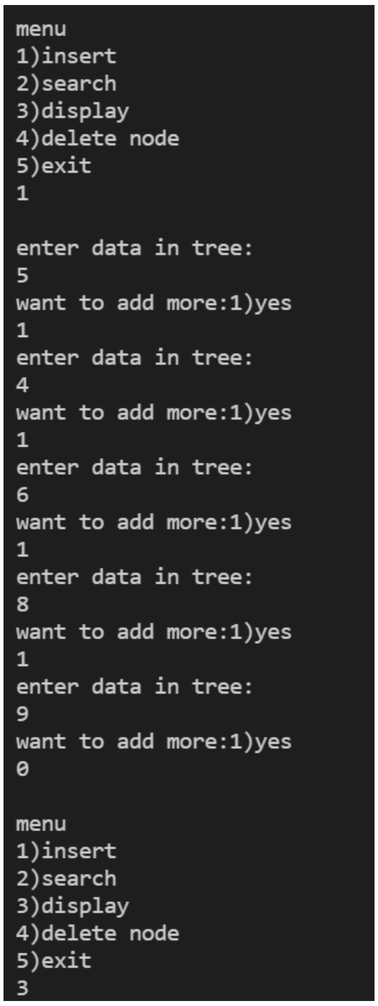
* Time complexity :-
The searching function of BST has a time complexity O(log n).

Conclusion :-
The concept and operations of binary search tree are implemented successfully.

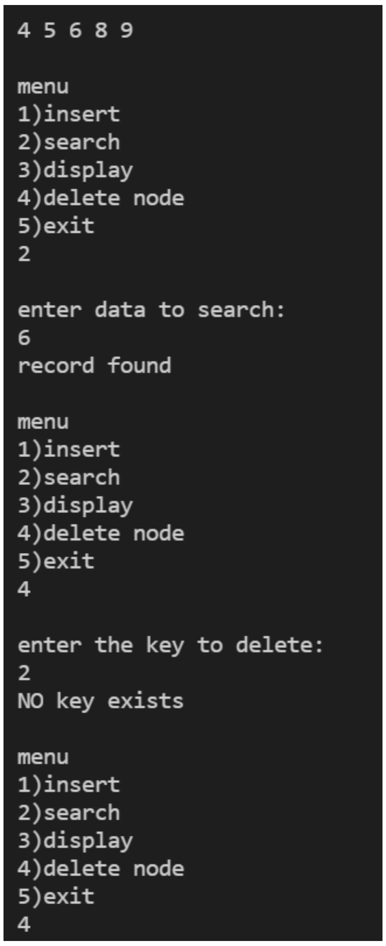
CODE:

```cpp
#include <iostream>
#include <iomanip>
using namespace std;
struct tnode
{
    int data;
    tnode *left = NULL, *right = NULL;
};
class tree
{
public:
    tnode *root = NULL;
    void create();
    void search();
    void display_ascend();
    void inorder(tnode *q);

    tnode *deletenode(tnode *root, int key)
    {
        tnode *temp;
        if (root == NULL)
        {
            cout << "NO key exists" << endl;
            return NULL;
        }
        else if (key < root->data)
        {
            root->left = deletenode(root->left, key);
        }
        else if (root->data < key)
        {
            root->right = deletenode(root->right, key);
        }
        else
        {
            if (root->left == NULL && root->right == NULL)
            {
                temp = root;
                root = NULL;
                delete temp;
            }
            else if (root->right == NULL)
            {
                temp = root;
                root = root->left;
                temp->left = NULL;
                delete temp;
            }
            else if (root->left == NULL)
            {
                temp = root;
                root = root->right;
                temp->left = NULL;
                delete temp;
            }
            else
            {
                temp = root;
                temp = temp->right;
                while (temp->left != NULL)
                {
                    temp = temp->left;
                }
                root->data = temp->data;
                root->right = deletenode(root->right, temp->data);
            }
            return root;
        }
    }
};

void tree::create()
{
    tnode *nn = new tnode;
    tnode *cn, *parent;
    cn = root;
    cout << "enter data in tree:" << endl;
    cin >> nn->data;
    if (cn == NULL)
    {
        root = nn;
    }
    else
    {
        while (cn != NULL)
        {
            parent = cn;
            if (cn->data > nn->data)
            {
                cn = cn->left;
            }
            else
            {
                cn = cn->right;
            }
        }
        if (parent->data > nn->data)
        {
            parent->left = nn;
        }
        else
        {
            parent->right = nn;
        }
    }
}
void tree::display_ascend()
{
    tnode *cn = root;
    if (cn == NULL)
    {
        cout << "binary tree is not in existance" << endl;
    }
    else
    {
        inorder(cn);
    }
    cout << endl;
}
void tree::inorder(tnode *root)
{
    tnode *cn = root;
    if (cn != NULL)
    {
        inorder(cn->left);
        cout << cn->data << " ";
        inorder(cn->right);
    }
}
void tree::search()
{
    tnode *cn = root;
    int m;
    cout << "enter data to search:" << endl;
    cin >> m;
    while (cn != NULL)
    {
        if (cn->data == m)
        {
            cout << "record found" << endl;
            break;
        }
        else if (cn->left != NULL || cn->right != NULL)
        {
            if (cn->data > m)
            {
                cn = cn->left;
            }
            else
            {
                cn = cn->right;
            }
        }
        else
        {
            cn = NULL;
        }
    }
    if (cn == NULL)
    {
        cout << "record not found" << endl;
    }
}

int main()
{
    tree t;
    int choice, key, ch;
    while (1)
    {
        cout << endl
             << "menu" << endl
             << "1)insert" << endl
             << "2)search" << endl
             << "3)display" << endl
             << "4)delete node" << endl
             << "5)exit" << endl;
        cin >> choice;
        cout << endl;
        switch (choice)
        {
        case 1:
            do
            {
                t.create();
                cout << "want to add more:1)yes" << endl;
                cin >> ch;
            } while (ch == 1);
            break;
        case 2:
            t.search();
            break;
        case 3:
            t.display_ascend();
            break;
        case 4:
            cout << "enter the key to delete:" << endl;
            cin >> key;
            t.deletenode(t.root, key);
            break;
        case 5:
            return 0;
            break;
        default:
            cout << "wrong choice" << endl;
            break;
        }
    }
    return 0;
}
```

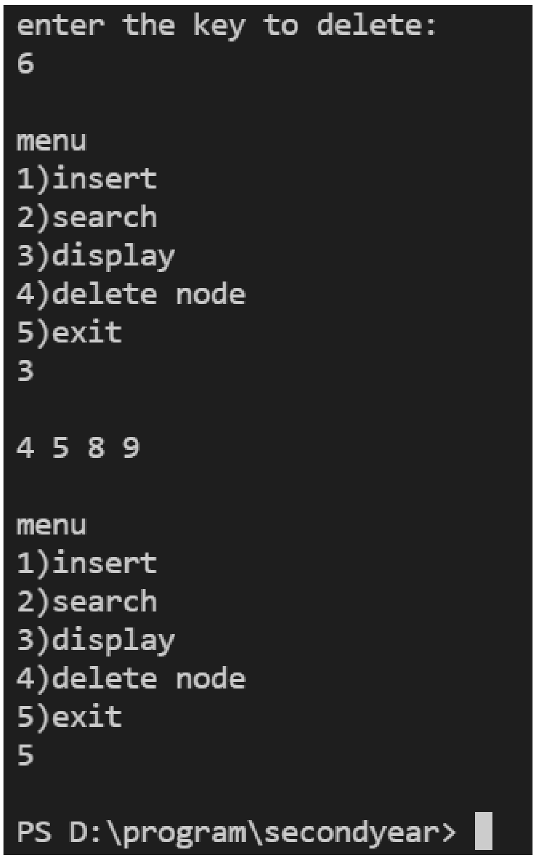
OUTPUT:



```
menu
1)insert
2)search
3)display
4)delete node
5)exit
1

enter data in tree:
5
want to add more:1)yes
1
enter data in tree:
4
want to add more:1)yes
1
enter data in tree:
6
want to add more:1)yes
1
enter data in tree:
8
want to add more:1)yes
1
enter data in tree:
9
want to add more:1)yes
0

menu
1)insert
2)search
3)display
4)delete node
5)exit
3
```



```
4 5 6 8 9

menu
1)insert
2)search
3)display
4)delete node
5)exit
2

enter data to search:
6
record found

menu
1)insert
2)search
3)display
4)delete node
5)exit
4

enter the key to delete:
2
NO key exists

menu
1)insert
2)search
3)display
4)delete node
5)exit
4
```



```
enter the key to delete:
6

menu
1)insert
2)search
3)display
4)delete node
5)exit
3

4 5 8 9

menu
1)insert
2)search
3)display
4)delete node
5)exit
5

PS D:\program\secondyear>
```