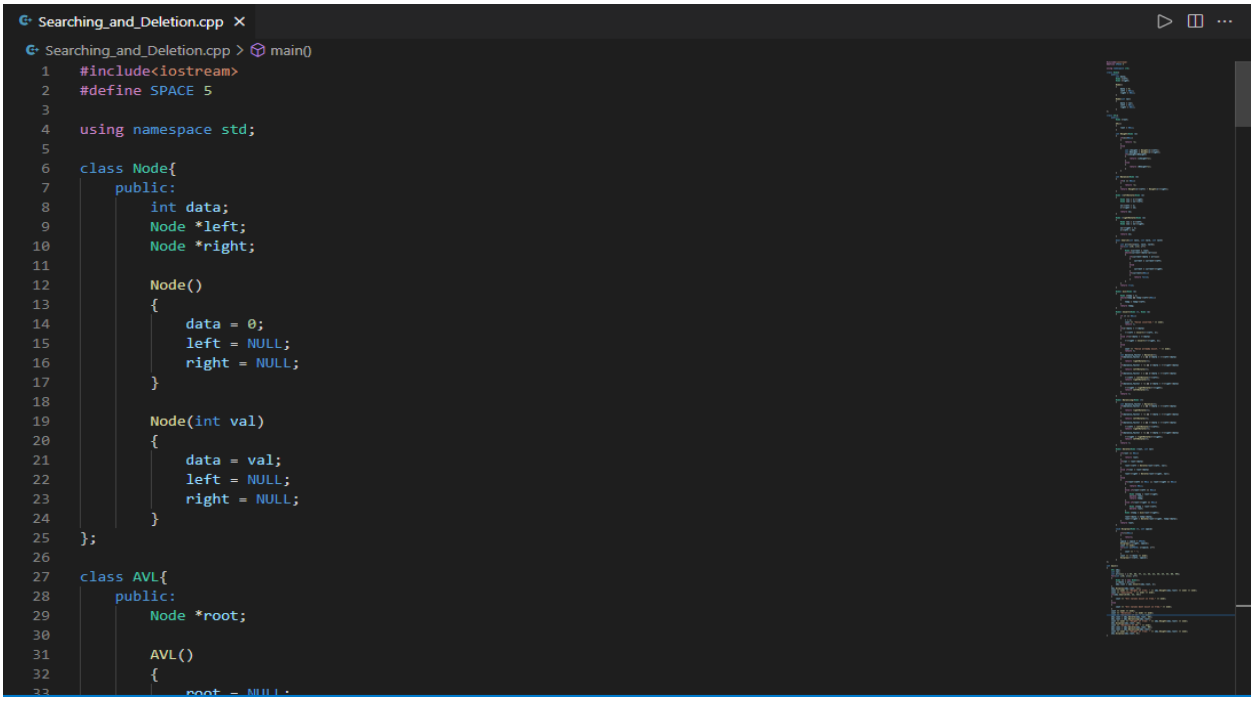
REMAINING LAB TASK # 10

MOHAMMAD BASIL ALI KHAN

20K-0477

Searching and Deletion

- **Code**



```cpp
#include<iostream>
#define SPACE 5

using namespace std;

class Node{
    public:
        int data;
        Node *left;
        Node *right;

        Node()
        {
            data = 0;
            left = NULL;
            right = NULL;
        }

        Node(int val)
        {
            data = val;
            left = NULL;
            right = NULL;
        }
};

class AVL{
    public:
        Node *root;

        AVL()
        {
            root = NULL;
        }

        int Height(Node *n)
        {
            if(n==NULL)
            {
                return -1;
            }
            else
            {
                int Lheight = Height(n->left);
                int Rheight = Height(n->right);
                if(Lheight>Rheight)
                {
                    return (Lheight+1);
                }
                else
                {
                    return (Rheight+1);
                }
            }
        }

        int Balance(Node *n)
        {
            if(n == NULL)
            {
                return -1;
            }
            return Height(n->left) - Height(n->right);
        }
```

Searching_and_Deletion.cpp

Searching_and_Deletion.cpp > main()

```
64      }
65
66      Node *leftRotate(Node *n)
67      {
68          Node *n1 = n->right;
69          Node *n2 = n1->left;
70
71          n1->left = n;
72          n->right = n2;
73
74          return n1;
75      }
76
77      Node *rightRotate(Node *n)
78      {
79          Node *n1 = n->left;
80          Node *n2 = n1->right;
81
82          n1->right = n;
83          n->left = n2;
84
85          return n1;
86      }
87
88      bool Search(int val1, int val2, int val3)
89      {
90          int arr[3]={val1, val2, val3};
91          for(int i=0; i<3; i++)
92          {
93              Node *current = root;
94              while(current->data!=arr[i])
95              {
```

Searching_and_Deletion.cpp

Searching_and_Deletion.cpp > main()

```
95              {
96                  if(current->data > arr[i])
97                  {
98                      current = current->left;
99                  }
100                 else
101                 {
102                     current = current->right;
103                 }
104                 if(current==NULL)
105                 {
106                     return false;
107                 }
108             }
109         }
110         return true;
111     }
112
113     Node* min(Node *n)
114     {
115         Node *temp = n;
116         while(temp && temp->left!=NULL)
117         {
118             temp = temp->left;
119         }
120         return temp;
121     }
122
123     Node* Insert(Node *r, Node *n)
124     {
125         if (r == NULL)
126         {
```

Searching_and_Deletion.cpp

Searching_and_Deletion.cpp > main()

```cpp
        if (r == NULL)
        {
            r = n;
            cout << "Value inserted." << endl;
            return r;
        }
        if(n->data < r->data)
        {
            r->left = Insert(r->left, n);
        }
        else if(n->data > r->data)
        {
            r->right = Insert(r->right, n);
        }
        else
        {
            cout << "Value already exist. " << endl;
            return r;
        }
        int balance_factor = Balance(r);
        if(balance_factor > 1 && n->data < r->left->data)
        {
            return rightRotate(r);
        }
        if(balance_factor < -1 && n->data > r->right->data)
        {
            return leftRotate(r);
        }
        if(balance_factor > 1 && n->data > r->left->data)
        {
            r->left = leftRotate(r->left);
            return rightRotate(r);
        }
```

Searching_and_Deletion.cpp

Searching_and_Deletion.cpp > main()

```cpp
        }
        if(balance_factor < -1 && n->data < r->right->data)
        {
            r->right = rightRotate(r->right);
            return leftRotate(r);
        }
        return r;
    }

    Node* Balancing(Node *r)
    {
        int balance_factor = Balance(r);
        if(balance_factor > 1 && r->data < r->left->data)
        {
            return rightRotate(r);
        }
        if(balance_factor < -1 && r->data > r->right->data)
        {
            return leftRotate(r);
        }
        if(balance_factor > 1 && r->data > r->left->data)
        {
            r->left = leftRotate(r->left);
            return rightRotate(r);
        }
        if(balance_factor < -1 && r->data < r->right->data)
        {
            r->right = rightRotate(r->right);
            return leftRotate(r);
        }
        return r;
    }
```

Searching_and_Deletion.cpp

Searching_and_Deletion.cpp > main()

```
190         Node* Delete(Node *root, int val)
191         {
192             if(root == NULL)
193             {
194                 return root;
195             }
196             if(val < root->data)
197             {
198                 root->left = Delete(root->left, val);
199             }
200             else if(val > root->data)
201             {
202                 root->right = Delete(root->right, val);
203             }
204             else
205             {
206                 if(root->left == NULL || root->right == NULL)
207                 {
208                     return NULL;
209                 }
210                 else if(root->left == NULL)
211                 {
212                     Node *temp = root->right;
213                     delete root;
214                     return temp;
215                 }
216                 else if(root->right == NULL)
217                 {
218                     Node *temp = root->left;
219                     delete root;
220                 }
221                 Node *temp = min(root->right);
```

Searching_and_Deletion.cpp

Searching_and_Deletion.cpp > main()

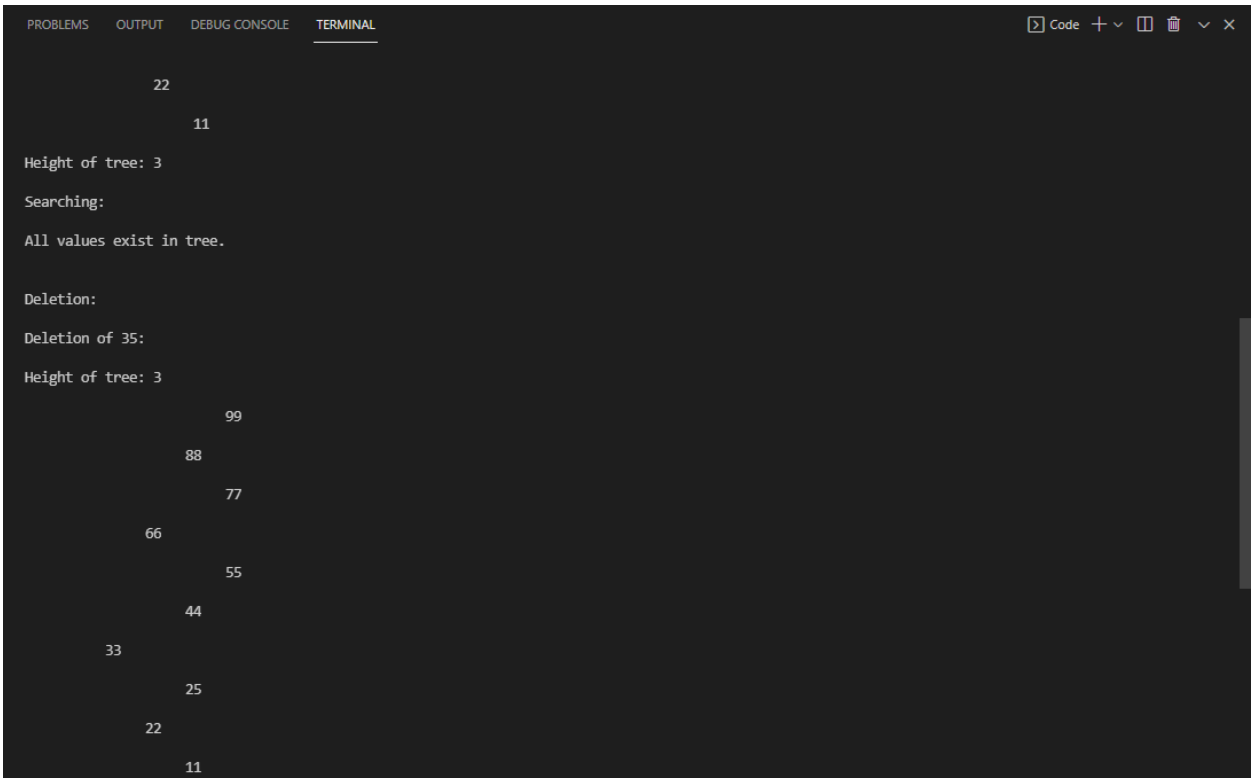
```
218                     Node *temp = root->left;
219                     delete root;
220                 }
221                 Node *temp = min(root->right);
222 
223                 root->data = temp->data;
224                 root->right = Delete(root->right, temp->data);
225             }
226             return root;
227         }
228 
229         void Display(Node *r, int space)
230         {
231             if(r==NULL)
232             {
233                 return;
234             }
235             space = space + SPACE;
236             Display(r->right, space);
237             cout << endl;
238             for(int i=SPACE; i<space; i++)
239             {
240                 cout << " ";
241             }
242             cout << r->data << endl;
243             Display(r->left, space);
244         }
245 };
246 
247 int main()
248 {
249     AVL obj;
250     int val;
```

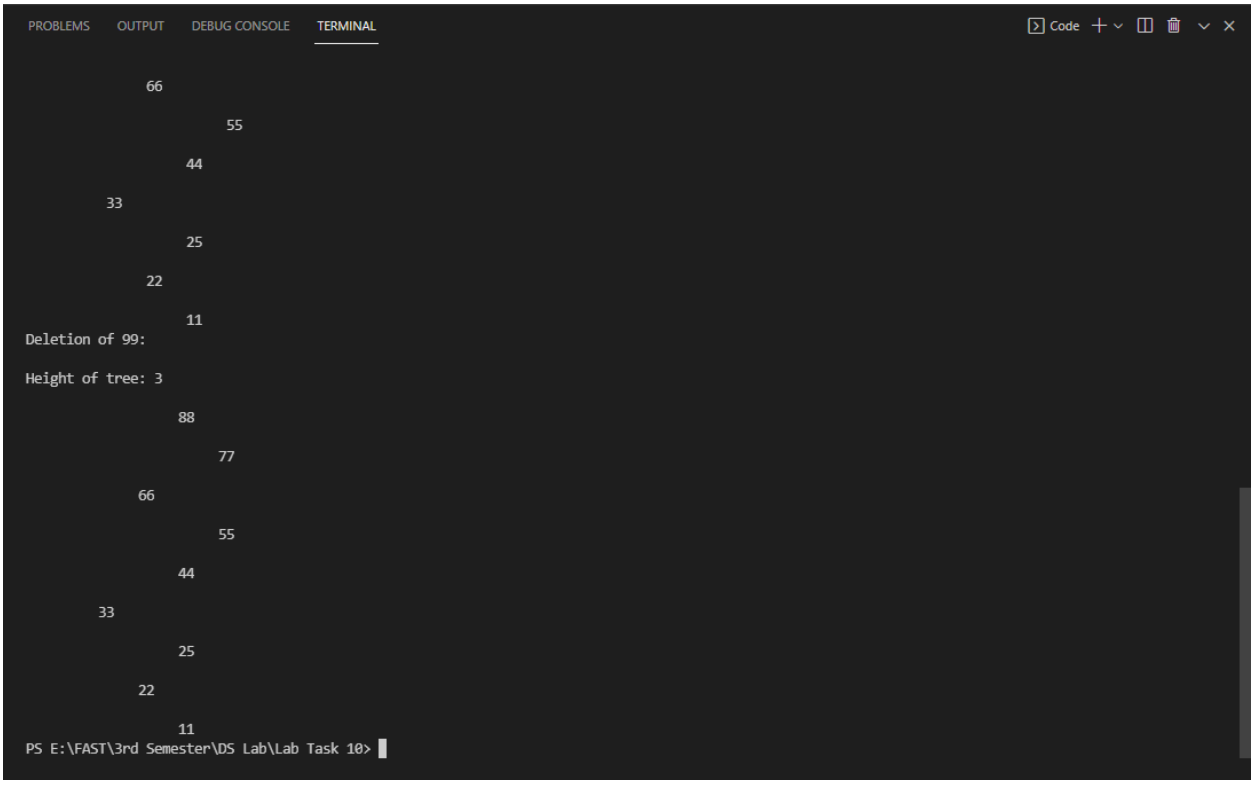
Searching_and_Deletion.cpp

```cpp
    int val;
    int arr[11] = { 55, 66, 77, 11, 33, 22, 35, 25, 44, 88, 99};
    for(int i=0; i<11; i++)
    {
        Node *n = new Node();
        n->data = arr[i];
        obj.root = obj.Insert(obj.root, n);
    }
    obj.Display(obj.root, 11);
    cout << endl << "Height of tree: " << obj.Height(obj.root) << endl << endl;
    cout << "Searching: " << endl << endl;
    if(obj.Search(66, 22, 44))
    {
        cout << "All values exist in tree." << endl;
    }
    else
    {
        cout << "All values dont exist in tree." << endl;
    }
    cout << endl << endl;
    cout << "Deletion: " << endl << endl;
    cout << "Deletion of 35: " << endl;
    obj.root = obj.Delete(obj.root, 35);
    obj.root = obj.Balancing(obj.root);
    cout << endl << "Height of tree: " << obj.Height(obj.root) << endl;
    obj.Display(obj.root, 10);
    cout << "Deletion of 99: " << endl;
    obj.root = obj.Delete(obj.root, 99);
    obj.root = obj.Balancing(obj.root);
    cout << endl << "Height of tree: " << obj.Height(obj.root) << endl;
    obj.Display(obj.root, 9);
}
```

- **Output**

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS E:\FAST\3rd Semester\DS Lab\Lab Task 10> cd "e:\FAST\3rd Semester\DS Lab\Lab Task 10\" ; if ($?) { g++ Searching_and_Deletion.cpp -o Searching_and_
Deletion } ; if ($?) { .\Searching_and_Deletion }
Value inserted.
Value inserted.
Value inserted.
Value inserted.
Value inserted.
Value inserted.
Value inserted.
Value inserted.
Value inserted.
Value inserted.
Value inserted.

                    99

                88

                    77

            66

                    55

                44

                    35
```


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
Code
22
11
Height of tree: 3
Searching:
All values exist in tree.
Deletion:
Deletion of 35:
Height of tree: 3
99
88
77
66
55
44
33
25
22
11


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
Code
66
55
44
33
25
22
11
Deletion of 99:
Height of tree: 3
88
77
66
55
44
33
25
22
11
PS E:\FAST\3rd Semester\DS Lab\Lab Task 10>