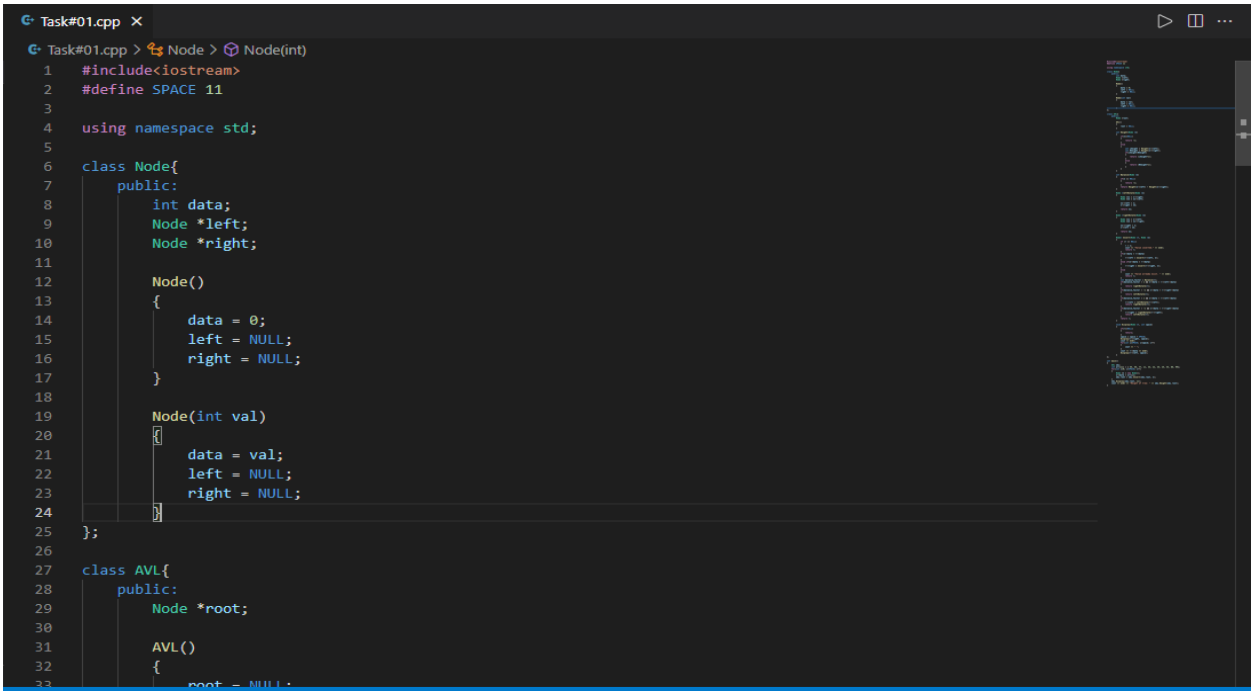
LAB TASK # 10

MOHAMMAD BASIL ALI KHAN

20K-0477

Task#01:

- **Code**



```cpp
#include<iostream>
#define SPACE 11

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;
            }
```

Task#01.cpp

Task#01.cpp > Node > Node(int)

```
61              return -1;
62          }
63          return Height(n->left) - Height(n->right);
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      Node* Insert(Node *r, Node *n)
89      {
90          if (r == NULL)
91          {
92              r = n;
```

Task#01.cpp

Task#01.cpp > Node > Node(int)

```
88      Node* Insert(Node *r, Node *n)
89      {
90          if (r == NULL)
91          {
92              r = n;
93              cout << "Value inserted." << endl;
94              return r;
95          }
96          if(n->data < r->data)
97          {
98              r->left = Insert(r->left, n);
99          }
100         else if(n->data > r->data)
101         {
102             r->right = Insert(r->right, n);
103         }
104         else
105         {
106             cout << "Value already exist. " << endl;
107             return r;
108         }
109         int balance_factor = Balance(r);
110         if(balance_factor > 1 && n->data < r->left->data)
111         {
112             return rightRotate(r);
113         }
114         if(balance_factor < -1 && n->data > r->right->data)
115         {
116             return leftRotate(r);
117         }
118         if(balance_factor > 1 && n->data > r->left->data)
119         {
```

Task#01.cpp

Task#01.cpp > Node > Node(int)

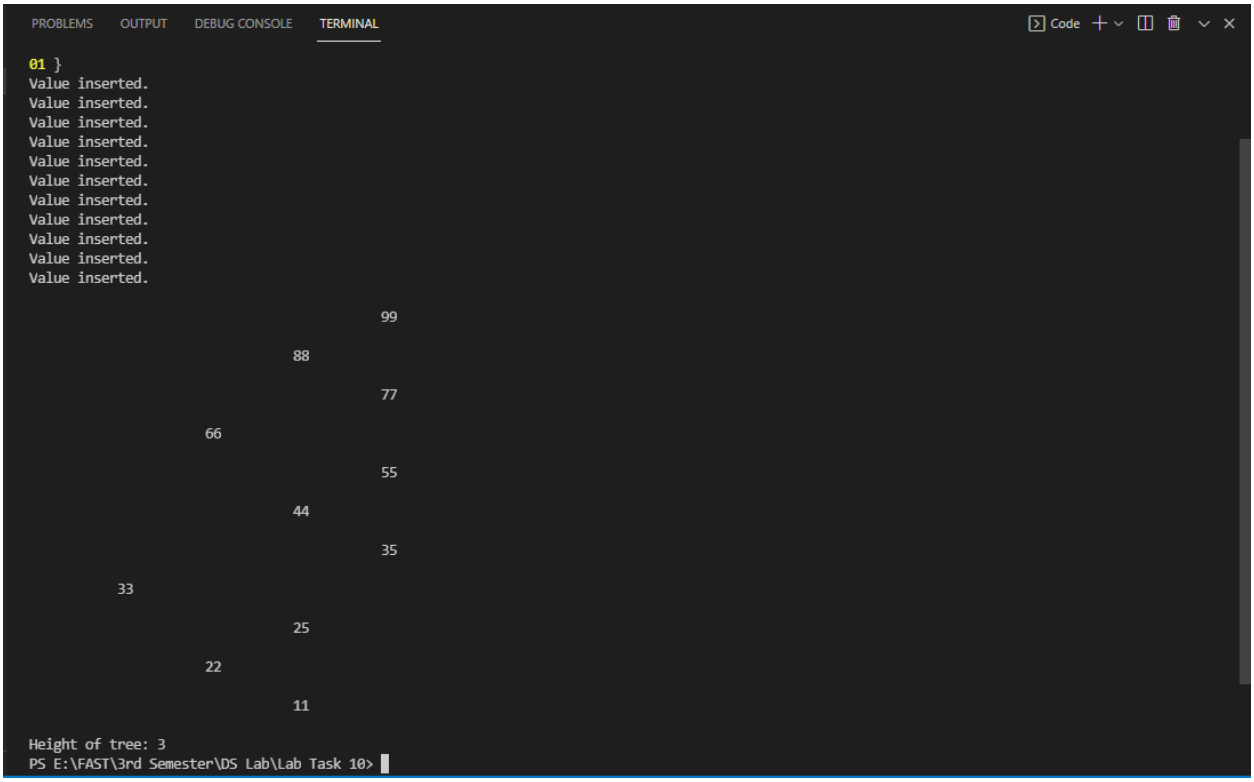
```
116                 return leftRotate(r);
117             }
118             if(balance_factor > 1 && n->data > r->left->data)
119             {
120                 r->left = leftRotate(r->left);
121                 return rightRotate(r);
122             }
123             if(balance_factor < -1 && n->data < r->right->data)
124             {
125                 r->right = rightRotate(r->right);
126                 return leftRotate(r);
127             }
128             return r;
129         }
130 
131         void Display(Node *r, int space)
132         {
133             if(r==NULL)
134             {
135                 return;
136             }
137             space = space + SPACE;
138             Display(r->right, space);
139             cout << endl;
140             for(int i=SPACE; i<space; i++)
141             {
142                 cout << " ";
143             }
144             cout << r->data << endl;
145             Display(r->left, space);
146         }
147 };
```

Task#01.cpp

Task#01.cpp > Node > Node(int)

```
130 
131         void Display(Node *r, int space)
132         {
133             if(r==NULL)
134             {
135                 return;
136             }
137             space = space + SPACE;
138             Display(r->right, space);
139             cout << endl;
140             for(int i=SPACE; i<space; i++)
141             {
142                 cout << " ";
143             }
144             cout << r->data << endl;
145             Display(r->left, space);
146         }
147 };
148 
149 int main()
150 {
151     AVL obj;
152     int arr[11] = { 55, 66, 77, 11, 33, 22, 35, 25, 44, 88, 99};
153     for(int i=0; i<SPACE; i++)
154     {
155         Node *n = new Node();
156         n->data = arr[i];
157         obj.root = obj.Insert(obj.root, n);
158     }
159     obj.Display(obj.root, 11);
160     cout << endl << "Height of tree: " << obj.Height(obj.root);
161 }
```

- **Ouput**


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
Code
01 }
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
33
25
22
11
Height of tree: 3
PS E:\FAST\3rd Semester\DS Lab\Lab Task 10>