Name : Dev Adnani
SID : 202212012
Subject : DSA
Topic : Binary Trees
Lab :06

```
#include <iostream>
using namespace std;

class Node
{

public:
	int data;

	Node *left, *right;

	Node(int data)
	{

	this->data = data;

	this->left = this->right = NULL;
	}
};

class BSTree
{

public:
	Node *root;

	int count;
```

```
BSTree()
{

this->root = NULL;

this->count = 0;
}

void insert(int data)
{

this->count++;

if (this->root == NULL)
{

        Node *newNode = new Node(data);

        this->root = newNode;

        return;
}

insert(this->root, data);
}

void insert(Node *node, int data)
{
```

```
if (node->data > data)
{

        if (node->left == NULL)
        {

        node->left = new Node(data);

        return;
        }

        insert(node->left, data);
}

else
{

        if (node->right == NULL)
        {

        node->right = new Node(data);

        return;
        }

        insert(node->right, data);
}
```

```
}

int maxFind(int a, int b)
{

if (a > b)
{

        return a;
}

if (b > a)
{

        return b;
}
}

int MaxHeight()
{

return this->MaxHeight(this->root);
}

int MaxHeight(Node *node)
{

if (node == NULL)
```

```
{

        return -1;
}

int leftMax = MaxHeight(node->left) + 1;

int rightMax = MaxHeight(node->right) + 1;

return maxFind(leftMax, rightMax);
}

bool search(int key)
{

return search(this->root, key);
}

bool search(Node *node, int key)
{

if (node == NULL)
{

        return 0;
}

if (node->data == key)
```

```
{

        return 1;
}

if (key > node->data)
{

        return search(node->right, key);
}

else
{

        return search(node->left, key);
}
}

void Inorder(Node *node)
{

if (node == NULL)
{

        return;
}

Inorder(node->left);
```

```
cout << node->data << " ";

Inorder(node->right);
}

int minValueTree(Node *node)
{

Node *temp = node;

while (temp != NULL && temp->left != NULL)
{

        temp = temp->left;
}

return temp->data;
}

void dNode(int key)
{

this->root = dNode(this->root, key);

this->count--;
}
```

```
Node *dNode(Node *node, int key)
{

if (node == NULL)
{

	return NULL;
}

if (key > node->data)
{

	cout << "to right from: " << node->data << endl;

	node->right = dNode(node->right, key);

	return node;
}

else if (key < node->data)
{

	cout << "to left from: " << node->data << endl;

	node->left = dNode(node->left, key);

	return node;
}
```

```
else
{

        if (node->left == NULL && node->right == NULL)
        {
        cout << "Found:- " << node->data << endl;
        return NULL;
        }

        else if (node->left == NULL)
        {

        Node *temp = node->right;

        free(node);

        return temp;
        }

        else if (node->right == NULL)
        {

        Node *temp = node->left;

        free(node);

        return temp;
```

```
        }

        cout << endl
        << "Min value found:" <<

        minValueTree(node->right) << endl;

        int minValue = minValueTree(node->right);

        node->data = minValue;

        node->right = dNode(node->right, minValue);

        return node;
}
}

int findMedian()
{

int middle = {0};

if (this->count % 2 == 0)
{

        middle = (this->count / 2 + ((this->count / 2) + 1)) / 2;
}
```

```
else
{

        middle = (this->count + 1) / 2;
}

return findMedian(this->root, middle);
}

int findMedian(Node *node, int middle)
{

if (node == NULL)
{

        return -1;
}

int temp = findMedian(node->left, middle) + 1;

if (temp == middle)
{

        return node->left->data;
}

temp++;
```

```
if (temp == middle)
{

	return node->data;
}

temp = findMedian(node->right, middle) + 1;

if (temp == middle)
{

	return node->data;
}
}

void findCommonAncestor(int key1, int key2)
{

bool found = false;

findCommonAncestor(this->root, key1, key2, found);

if (found == false)
{

	cout << "Common lowser ancestor not found.";
}
}
```

```
bool findCommonAncestor(Node *node, int key1, int key2, bool &found)

{

bool common, left, right;

if (node == NULL)
{

    return false;
}

if (node->data == key1 && node->data == key2)
{

    cout << "Common lowest ancestor is:" << node->data << endl;

    found = true;

    return false;
}

else if (node->data == key1 || node->data == key2)
{
    common = true;
```

```
    }

    else
    {

        common = false;
    }

    left = findCommonAncestor(node->left, key1, key2,
found);
    right = findCommonAncestor(node->right, key1, key2,
found);

    if (common == true)
    {

        if (left == true || right == true)
        {

        cout << "Common lowest ancestor is : " <<
node->data << endl;

        found = true;

        return false;
        }

        else
```

```
            {

            return true;
            }
        }

        if (left == true && right == true)
        {

            cout << "Q5 : Common lowest ancestor is:" <<
node->data << endl;

            found = true;

            return false;
        }

        else if (left == true || right == true)
        {

            return true;
        }

        else
        {

            return false;
        }
```

```
        }
};

int main()
{

        BSTree bst;

        bst.insert(1);
        bst.insert(2);
        bst.insert(3);
        bst.insert(4);
        bst.insert(5);
        bst.insert(6);
        bst.insert(7);
        bst.insert(8);
        bst.insert(9);
        bst.insert(10);
        bst.insert(11);
        bst.insert(12);



        // Q1
        cout << "Q1 : Height of the tree: " << bst.MaxHeight() <<
endl;
        cout<<endl;
```

```
	// Q2
	int find = 7;
	bool found = bst.search(find);
	if (found)
	{
	cout << "Q2 : Found data: " << find << endl;
	}
	else
	{
	cout << "Q2 : Not found data: " << find << endl;
	}
	cout<<endl;

	//Q3
	cout<<"Q3 :"<<endl;
	bst.Inorder(bst.root);
	cout << endl;
	bst.dNode(5);
	bst.Inorder(bst.root);
	cout << endl;
	cout << endl;


	//Q4
	cout<<"Q4 : "<<endl;
	bst.Inorder(bst.root);
	cout << endl<<"Median is: " << bst.findMedian() << endl
<<endl;
```

```
	//Q5
	bst.findCommonAncestor(11,1);

	return 0;
}
```

O/P :

```
devadnani@Devs-MacBook-Pro Misc % cd "/Users/devadnani/Desktop/Misc"
devadnani@Devs-MacBook-Pro Misc % cd "/Users/devadnani/Desktop/Misc/" && g++
d.cpp -o d && "/Users/devadnani/Desktop/Misc/"d
d.cpp: In member function 'int BSTree::maxFind(int, int)':
d.cpp:102:5: warning: control reaches end of non-void function [-Wreturn-type]
  102 |     }
      |     ^
d.cpp: In member function 'int BSTree::findMedian(Node*, int)':
d.cpp:323:5: warning: control reaches end of non-void function [-Wreturn-type]
  323 |     }
      |     ^
Q1 : Height of the tree: 11

Q2 : Found data: 7

Q3 :
1 2 3 4 5 6 7 8 9 10 11 12
to right from: 1
to right from: 2
to right from: 3
to right from: 4
1 2 3 4 6 7 8 9 10 11 12

Q4 :
1 2 3 4 6 7 8 9 10 11 12
Median is: 6

Common lowest ancestor is  :  1
devadnani@Devs-MacBook-Pro Misc %
```