

ASSIGNMENT 6

```
#include <bits/stdc++.h>

#include <string>

using namespace std;

class BSTNode {
private:
    int data;
    BSTNode* left;
    BSTNode* right;

public:
    BSTNode(int d): data(d), left(NULL), right(NULL) {}

    friend class BST;
};

class BST {
private:
    BSTNode* root;

public:
    BST(): root(NULL) {}

    void insert(int data) {
        BSTNode* temp = root;
        BSTNode* prev = root;
        if (!root) {
            root = new BSTNode(data);
            root->data = data;
```

```

        return;
    }

    while (temp) {
        prev = temp;
        if (data > temp->data) {
            temp = temp->right;
        } else {
            temp = temp->left;
        }
    }

    if (data > prev->data) {
        prev->right = new BSTNode(data);
    } else {
        prev->left = new BSTNode(data);
    }
}

int min() {
    if (!root) {
        return 0;
    }

    BSTNode* temp = this->root;
    while (temp->left != NULL)
        temp = temp->left;

    return temp->data;
}

```

```

void swapLeftRight() {
    if (!root) {
        return;
    }

    queue<BSTNode*> level;

    BSTNode* curr;
    BSTNode* temp;
    level.push(root);

    while (!level.empty()) {
        curr = level.front();
        level.pop();

        temp = curr->left;
        curr->left = curr->right;
        curr->right = temp;

        if (curr->left) {
            level.push(curr->left);
        }
        if (curr->right) {
            level.push(curr->right);
        }
    }
}

```

```

bool search(int value) {
    if (!root) {
        return false;
    }
}

```

```
}
```

```
BSTNode* temp = root;
```

```
while (temp) {
```

```
    if (value == temp->data) {
```

```
        return true;
```

```
    }
```

```
    else if (value < temp->data) {
```

```
        temp = temp->left;
```

```
    }
```

```
    else if (value > temp->data) {
```

```
        temp = temp->right;
```

```
    }
```

```
}
```

```
return false;
```

```
}
```

```
int height() {
```

```
    if (!root) {
```

```
        return 0;
```

```
    }
```

```
queue<BSTNode*> level;
```

```
int height = 0;
```

```
unsigned int nodes;
```

```
BSTNode* temp;
```

```
level.push(this->root);
```

```

/*
 * Adds child nodes to current level
 * Repeat the procedure for each item in the level
 */
while(!level.empty()) {
    height++;
    nodes = level.size();
    while (nodes--) {
        temp = level.front();
        level.pop();
        if (temp->left) {
            level.push(temp->left);
        }
        if (temp->right) {
            level.push(temp->right);
        }
    }
}

return height;
}

```

```

void display() {
    if (!root) {
        cout << "[]" << endl;
        return;
    }

    cout << "ARRAY REPRESENTATION: [ " << flush;

    queue<BSTNode*> level;

```

```

vector<string> representation;

level.push(root);

representation.push_back(to_string(root->data));


BSTNode* temp;
while(!level.empty()) {
    temp = level.front();
    level.pop();

    if (temp->left) {
        representation.push_back(to_string(temp->left->data));
        level.push(temp->left);
    } else {
        representation.emplace_back("NULL");
    }

    if (temp->right) {
        representation.push_back(to_string(temp->right->data));
        level.push(temp->right);
    } else {
        representation.emplace_back("NULL");
    }
}

for (auto& item : representation) {
    cout << item << ", ";
}
cout << "]" << endl;
}
};

```

```

int main() {

    BST bst;

    /*
    *   CONSTRUCT FOLLOWING BINARY TREE:
    *
    *       10
    *      / \
    *     2  15
    *    /\   \
    *   1 3   20
    *      /
    *     16
    */
    bst.insert(10);
    bst.insert(15);
    bst.insert(2);
    bst.insert(1);
    bst.insert(20);
    bst.insert(3);
    bst.insert(16);

    cout << "Height of tree: " << bst.height() << endl;
    cout << "Minimum element in tree: " << bst.min() << endl;

    if (bst.search(15)) {
        cout << "15 is present in the binary tree" << endl;
    } else {
        cout << "15 is not present in the binary tree" << endl;
    }

    if (bst.search(17)) {

```

```

        cout << "17 is present in the binary tree" << endl;
    } else {
        cout << "17 is not present in the binary tree" << endl;
    }

    bst.display();
    bst.swapLeftRight();
    bst.display();

    return 0;
}

```

OUTPUT :

The screenshot shows a web browser window with the URL https://www.onlinegdb.com/online_c++_compiler. The left sidebar contains navigation links like 'IDE', 'My Projects', 'Classroom', 'Learn Programming', 'Programming Questions', 'Jobs', 'Sign Up', 'Login', and 'Learn Python with KodeKloud'. The main editor area shows a C++ file named 'main.cpp' with the following code:

```

200 bst.insert(2);
201 bst.insert(1);
202 bst.insert(20);
203 bst.insert(3);
204 bst.insert(16);
205
206 cout << "Height of tree: " << bst.height() << endl;
207 cout << "Minimum element in tree: " << bst.min() << endl;
208
209 if (bst.search(15)) {
210     cout << "15 is present in the binary tree" << endl;
211 } else {
212     cout << "15 is not present in the binary tree" << endl;
213 }
214
215 if (bst.search(17)) {
216     cout << "17 is present in the binary tree" << endl;
217 } else {
218     cout << "17 is not present in the binary tree" << endl;
219 }
220

```

The output console at the bottom shows the following results:

```

Height of tree: 4
Minimum element in tree: 1
15 is present in the binary tree
17 is not present in the binary tree
ARRAY REPRESENTATION: [ 10, 2, 15, 1, 3, NULL, 20, NULL, NULL, NULL, NULL, 16, NULL, NULL, NULL, ]
ARRAY REPRESENTATION: [ 10, 15, 2, 20, NULL, 3, 1, NULL, 16, NULL, NULL, NULL, NULL, NULL, NULL, ]
...Program finished with exit code 0
Press ENTER to exit console.

```