```
‘’’ Name -Mahesh Gaikwad

Roll No. SA21

Problem Statement -Beginning with an empty binary search tree, Construct a binary search tree

by inserting the values in the order given. After constructing a binary tree -i. Insert new node, ii.

 Find number of nodes in longest path from root, iii. Minimum data value found in the tree, iv.

 Change a tree so that the roles of the left and right pointers are swapped at every node, v. Search a
value’’’

#include <iostream>

using namespace std;

struct Node { int data;

Node* left;

Node* right;

};

Node* insert(Node* root, int value) {

if (root == nullptr) { root = new

Node; root->data = value; root->left

= root->right =NULL;

}

if (value < root->data) { root->left = insert(root-

>left, value); } else if (value

> root->data) { root->right = insert(root->right,

value);

}

return root;

}

int findMin(Node* root) { while

(root->left != nullptr) { root =

root->left;
}
return root->data;
}

void displayInOrder(Node* root)
{ if (root != nullptr) {
displayInOrder(root->left); cout <<
root->data << "->";
displayInOrder(root->right);
}
}

bool search(Node* root, int value) {

if (root == nullptr) { return
false;
}
if (value == root->data) { return true;
} else if (value < root->data)
{ return search(root->left, value); }
else { return search(root->right,
value);
}
}
int main() { Node* root
= nullptr; int
numNodes;
```

```
cout << "Enter the number of nodes to insert: "; cin
>> numNodes; cout << "Enter the values to
insert, separated by spaces:\n"; for (int i = 0; i <
numNodes; ++i) { int value; cin >> value; root =
insert(root, value);
}
cout << "Elements in the tree (in-order traversal):
"; displayInOrder(root); cout
<< "\n";

int minValue = findMin(root); cout << "Minimum data
value in the tree: " << minValue << "\n"; int
searchValue; cout << "Enter a value to search: "; cin
>> searchValue; if (search(root, searchValue)) { cout
<< "Value " << searchValue << " found in the tree.\n";
} else { cout << "Value " << searchValue << " not
found in the tree.\n"; } cout << "Mirroring the
tree...\n"; mirrorTree(root); cout << "Elements in the
tree after mirroring (in-order traversal): ";
displayInOrder(root); cout << "\n"; return 0;
}
```

OUTPUT:

```
Enter the number of nodes to insert: 8
Enter the values to insert, separated by spaces:
2 34 45 12 5 1 4 16
Elements in the tree (in-order traversal):
1->2->4->5->12->16->34->45->
```

```
Minimum data value in the tree: 1

Height of the tree : -3 Enter a

value to search: 4

Value 4 found in the tree.

Mirroring the tree...

Elements in the tree after

mirroring (in-order

traversal): 1->0->
```