**GROUP B**

**Beginning with an empty binary search tree, construct 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 everynode**

**v. Search a value**

#include <iostream>

#include <cstdlib>

using namespace std;

// Node structure for binary search tree

struct Node {

int data;

Node\* left;

Node\* right;

Node(int value) : data(value), left(nullptr), right(nullptr) {}

};

// Function to insert a new node into a binary search tree

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

if (root == nullptr) {

return new Node(value);

}

if (value < root->data) {

root->left = insert(root->left, value);

} else {

root->right = insert(root->right, value);

}

return root;

}

// Function to find the number of nodes in the longest path from the root

int findLongestPath(Node\* root) {

if (root == nullptr) {

return 0;

}

int leftPath = findLongestPath(root->left);

int rightPath = findLongestPath(root->right);

return max(leftPath, rightPath) + 1;

}

// Function to find the minimum data value in a binary search tree

int findMinValue(Node\* root) {

if (root->left == nullptr) {

return root->data;

}

return findMinValue(root->left);

}

// Function to swap the left and right pointers of all nodes in a binary tree

void swapPointers(Node\* root) {

if (root == nullptr) {

return;

}

swapPointers(root->left);

swapPointers(root->right);

swap(root->left, root->right);

}

// Function to search for a value in a binary search tree

bool searchValue(Node\* root, int value) {

if (root == nullptr) {

return false;

}

if (root->data == value) {

return true;

} else if (value < root->data) {

return searchValue(root->left, value);

} else {

return searchValue(root->right, value);

}

}

// Function to perform in-order traversal of a binary search tree

void inorderTraversal(Node\* root) {

if (root == nullptr) {

return;

}

inorderTraversal(root->left);

cout << root->data << " ";

inorderTraversal(root->right);

}

int main() {

Node\* root = nullptr;

int choice;

while (true) {

cout << "\nBinary Search Tree Menu:\n";

cout << "1. Insert a value\n";

cout << "2. Find the number of nodes in the longest path from the root\n";

cout << "3. Find the minimum data value in the tree\n";

cout << "4. Swap the left and right pointers of all nodes\n";

cout << "5. Search for a value\n";

cout << "6. Print the tree (in-order traversal)\n";

cout << "7. Exit\n";

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case 1: {

int value;

cout << "Enter the value to insert: ";

cin >> value;

root = insert(root, value);

cout << "Value inserted successfully.\n";

break;

}

case 2: {

int longestPath = findLongestPath(root);

cout << "Number of nodes in the longest path from the root: " << longestPath << endl;

break;

}

case 3: {

int minValue = findMinValue(root);

cout << "Minimum data value found in the tree: " << minValue << endl;

break;

}

case 4: {

swapPointers(root);

cout << "Pointers swapped successfully.\n";

break;

}

case 5: {

int value;

cout << "Enter the value to search: ";

cin >> value;

bool found = searchValue(root, value);

if (found) {

cout << "Value " << value << " found in the tree.\n";

} else {

cout << "Value " << value << " not found in the tree.\n";

}

break;

}

case 6: {

cout << "Binary Search Tree (in-order traversal): ";

inorderTraversal(root);

cout << endl;

break;

}

case 7: {

cout << "Exiting...\n";

exit(0);

}

default:

cout << "Invalid choice. Please try again.\n";

break;

}

}

}

OUTPUT:-

Binary Search Tree Menu:

1. Insert a value

2. Find the number of nodes in the longest path from the root

3. Find the minimum data value in the tree

4. Swap the left and right pointers of all nodes

5. Search for a value

6. Print the tree (in-order traversal)

7. Exit

Enter your choice: 1

Enter the value to insert: 2

Value inserted successfully.

Binary Search Tree Menu:

1. Insert a value

2. Find the number of nodes in the longest path from the root

3. Find the minimum data value in the tree

4. Swap the left and right pointers of all nodes

5. Search for a value

6. Print the tree (in-order traversal)

7. Exit

Enter your choice: 2

Number of nodes in the longest path from the root: 1

Binary Search Tree Menu:

1. Insert a value

2. Find the number of nodes in the longest path from the root

3. Find the minimum data value in the tree

4. Swap the left and right pointers of all nodes

5. Search for a value

6. Print the tree (in-order traversal)

7. Exit

Enter your choice: 3

Minimum data value found in the tree: 2

Binary Search Tree Menu:

1. Insert a value

2. Find the number of nodes in the longest path from the root

3. Find the minimum data value in the tree

4. Swap the left and right pointers of all nodes

5. Search for a value

6. Print the tree (in-order traversal)

7. Exit

Enter your choice: 4

Pointers swapped successfully.

Binary Search Tree Menu:

1. Insert a value

2. Find the number of nodes in the longest path from the root

3. Find the minimum data value in the tree

4. Swap the left and right pointers of all nodes

5. Search for a value

6. Print the tree (in-order traversal)

7. Exit

Enter your choice: 5

Enter the value to search: 2

Value 2 found in the tree.

Binary Search Tree Menu:

1. Insert a value

2. Find the number of nodes in the longest path from the root

3. Find the minimum data value in the tree

4. Swap the left and right pointers of all nodes

5. Search for a value

6. Print the tree (in-order traversal)

7. Exit

Enter your choice: 6

Binary Search Tree (in-order traversal): 2

Binary Search Tree Menu:

1. Insert a value

2. Find the number of nodes in the longest path from the root

3. Find the minimum data value in the tree

4. Swap the left and right pointers of all nodes

5. Search for a value

6. Print the tree (in-order traversal)

7. Exit

Enter your choice: 2

Number of nodes in the longest path from the root: 1

Binary Search Tree Menu:

1. Insert a value

2. Find the number of nodes in the longest path from the root

3. Find the minimum data value in the tree

4. Swap the left and right pointers of all nodes

5. Search for a value

6. Print the tree (in-order traversal)

7. Exit

Enter your choice: 1

Enter the value to insert: 33

Value inserted successfully.

Binary Search Tree Menu:

1. Insert a value

2. Find the number of nodes in the longest path from the root

3. Find the minimum data value in the tree

4. Swap the left and right pointers of all nodes

5. Search for a value

6. Print the tree (in-order traversal)

7. Exit

Enter your choice: 6

Binary Search Tree (in-order traversal): 2 33

Binary Search Tree Menu:

1. Insert a value

2. Find the number of nodes in the longest path from the root

3. Find the minimum data value in the tree

4. Swap the left and right pointers of all nodes

5. Search for a value

6. Print the tree (in-order traversal)

7. Exit

Enter your choice: 7

Exiting...