**PRACTICAL ASSIGNMENT::5**

**GITHUB:**[**Aditya6768/DSAL: all LAB work 2024-25 (github.com)**](https://github.com/Aditya6768/DSAL)

**Implement binary search tree and perform following operations:**

**a) Insert (Handle insertion of duplicate entry)**

**b) Delete**

**c) Search**

**d) Display tree (Traversal)**

#include <bits/stdc++.h>

using namespace std;

struct Node {

int key;

Node \*left;

Node \*right;

Node(int item) : key(item), left(NULL), right(NULL) {}

};

Node\* insert(Node\* node, int key) {

if (node == NULL)

return new Node(key);

if (key < node->key)

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

else if (key > node->key)

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

return node;

}

Node\* search(Node\* root, int key) {

if (root == NULL || root->key == key)

return root;

if (key < root->key)

return search(root->left, key);

return search(root->right, key);

}

Node\* findMin(Node\* node) {

while (node && node->left != NULL)

node = node->left;

return node;

}

Node\* deleteNode(Node\* root, int key) {

if (root == NULL)

return root;

if (key < root->key)

root->left = deleteNode(root->left, key);

else if (key > root->key)

root->right = deleteNode(root->right, key);

else {

if (root->left == NULL)

return root->right;

else if (root->right == NULL)

return root->left;

Node\* temp = findMin(root->right);

root->key = temp->key;

root->right = deleteNode(root->right, temp->key);

}

return root;

}

void inorder(Node\* root) {

if (root != NULL) {

inorder(root->left);

cout << root->key << " ";

inorder(root->right);

}

}

int main() {

Node\* root = NULL;

int choice, key;

while (true) {

cout << "\nMenu:\n";

cout << "1. Insert\n";

cout << "2. Delete\n";

cout << "3. Search\n";

cout << "4. Inorder Traversal\n";

cout << "5. Exit\n";

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case 1:

cout << "Enter key to insert: ";

cin >> key;

root = insert(root, key);

break;

case 2:

cout << "Enter key to delete: ";

cin >> key;

root = deleteNode(root, key);

break;

case 3:

cout << "Enter key to search: ";

cin >> key;

if (search(root, key))

cout << "Key " << key << " found in the BST.\n";

else

cout << "Key " << key << " not found in the BST.\n";

break;

case 4:

cout << "Inorder traversal: ";

inorder(root);

cout << "\n";

break;

case 5:

exit(0);

default:

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

}

}

return 0;

}

