**WEEK 9**

**BST**

#include <stdio.h>

#include <stdlib.h>

struct Node {

int data;

struct Node \*left, \*right;

};

// Function to create a new BST node

struct Node\* createNode(int data) {

struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node));

newNode->data = data;

newNode->left = newNode->right = NULL;

return newNode;

}

struct Node\* insertNode(struct Node\* root, int data) {

if (root == NULL) return createNode(data);

if (data < root->data)

root->left = insertNode(root->left, data);

else if (data > root->data)

root->right = insertNode(root->right, data);

return root;

}

struct Node\* findMin(struct Node\* node) {

struct Node\* current = node;

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

current = current->left;

return current;

}

struct Node\* deleteNode(struct Node\* root, int data) {

if (root == NULL) return root;

if (data < root->data)

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

else if (data > root->data)

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

else {

if (root->left == NULL) {

struct Node\* temp = root->right;

free(root);

return temp;

} else if (root->right == NULL) {

struct Node\* temp = root->left;

free(root);

return temp;

}

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

root->data = temp->data;

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

}

return root;

}

struct Node\* searchNode(struct Node\* root, int data) {

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

return root;

if (root->data < data)

return searchNode(root->right, data);

return searchNode(root->left, data);

}

void inOrder(struct Node\* root) {

if (root != NULL) {

inOrder(root->left);

printf("%d ", root->data);

inOrder(root->right);

}

}

int main() {

struct Node\* root = NULL;

int choice, data,n;

printf("Enter the no of elements to be inserted");

scanf("%d",&n);

printf("Enter elements");

for(int i=0;i<n;i++)

{ scanf("%d",&data);

root=insertNode(root, data);}

while (1) {

printf("\nBinary Search Tree Operations Menu\n");

printf("1. Insert\n");

printf("2. Delete\n");

printf("3. Search\n");

printf("4. Display\n");

printf("5. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

printf("Enter data to insert: ");

scanf("%d", &data);

root = insertNode(root, data);

printf("%d inserted.\n", data);

break;

case 2:

printf("Enter data to delete: ");

scanf("%d", &data);

root = deleteNode(root, data);

printf("%d deleted.\n", data);

break;

case 3:

printf("Enter data to search: ");

scanf("%d", &data);

struct Node\* foundNode = searchNode(root, data);

if (foundNode != NULL)

printf("%d found in the tree.\n", data);

else

printf("%d not found in the tree.\n", data);

break;

case 4:

printf("In-order display of the BST: ");

inOrder(root);

printf("\n");

break;

case 5:

exit(0);

break;

default:

printf("Invalid choice! Please try again.\n");

}

}

return 0;

}