**Binary Search Tree - Insertion, Deletion and Searching**

*Source Code:*

#include <stdio.h>

#include <stdlib.h>

struct Node {

    int data;

    struct Node\* left;

    struct Node\* right;

};

struct Node\* createNode(int data) {

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

    newNode->data = data;

    newNode->left = NULL;

    newNode->right = NULL;

    return newNode;

}

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

    if (root == NULL) {

    return createNode(data);

}

if (data < root->data) {

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

    }

else if (data > root->data) {

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

}

return root;

}

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

    while (node->left != NULL) {

        node = node->left;

    }

return node;

}

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

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

    return root;

}

    if (data < root->data) {

    return search(root->left, data);

    }

    return search(root->right, data);

}

void inOrderTraversal(struct Node\* root) {

if (root != NULL) {

inOrderTraversal(root->left);

printf("%d ", root->data);

inOrderTraversal(root->right);

}

}

void displayBST(struct Node\* root) {

printf("Updated Binary Search Tree: ");

inOrderTraversal(root);

printf("\n");

}

int main() {

    struct Node\* root = NULL;

    int choice, data;

    while (1) {

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

    printf("1. Insert\n2. Delete\n3. Search\n4. Display BST\n5. Exit\n");

    printf("Enter your choice: ");

    scanf("%d", &choice);

    switch (choice) {

        case 1:

        printf("Enter data to insert: ");

        scanf("%d", &data);

        root = insert(root, data);

        displayBST(root);

        break;

        case 2:

        printf("Enter data to delete: ");

        scanf("%d", &data);

        root = deleteNode(root, data);

        displayBST(root);

        break;

        case 3:

        printf("Enter data to search: ");

        scanf("%d", &data);

        if (search(root, data) != NULL) {

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

        } else {

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

        }

        break;

        case 4:

        displayBST(root);

        break;

        case 5:

        exit(0);

        default:

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

        }

    }

return 0;

}

*Output:*

Binary Search Tree Operations:

1. Insert

2. Delete

3. Search

4. Display BST

5. Exit

Enter your choice: 1

Enter data to insert: 1

Updated Binary Search Tree: 1

Binary Search Tree Operations:

1. Insert

2. Delete

3. Search

4. Display BST

5. Exit

Enter your choice: 1

Enter data to insert: 0

Updated Binary Search Tree: 0 1

Binary Search Tree Operations:

1. Insert

2. Delete

3. Search

4. Display BST

5. Exit

Enter your choice: 1

Enter data to insert: 2

Updated Binary Search Tree: 0 1 2

Binary Search Tree Operations:

1. Insert

2. Delete

3. Search

4. Display BST

5. Exit

Enter your choice: 2

Enter data to delete: 0

Updated Binary Search Tree: 1 2

Binary Search Tree Operations:

1. Insert

2. Delete

3. Search

4. Display BST

5. Exit

Enter your choice: 3

Enter data to search: 2

2 found in the tree.

Binary Search Tree Operations:

1. Insert

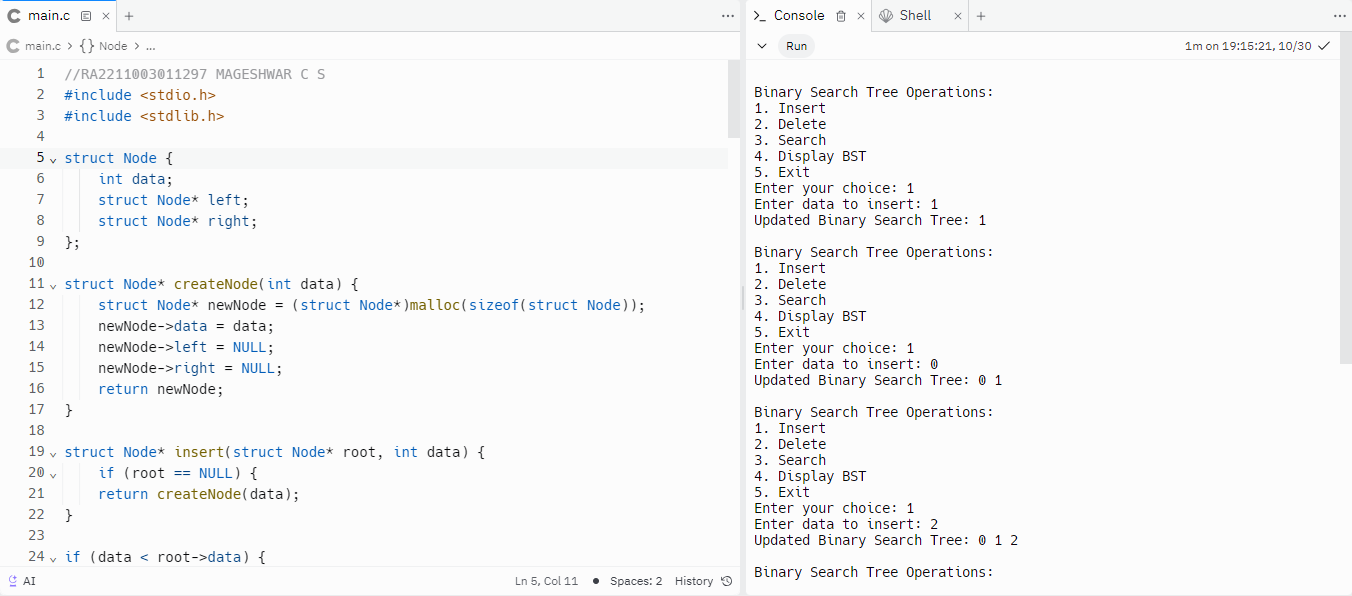
2. Delete

3. Search

4. Display BST

5. Exit

Enter your choice: 5



**ABISHEK S R  
RA2211003011292**