**NAME: Manasvi Mahadev Patil.**

**Branch: I.T.**

**Division: S. Y. - C**

**Roll no.:03**

**AIM:-**

**Write C/C++ program to Create BST, Take input user choice, Perform Insertion and deletion on BST. (Include all cases for both)**

The C++ program is designed to create a Binary Search Tree (BST) and allows the user to perform insertion and deletion operations on the tree. The program provides a user-friendly menu to interact with the BST and display its contents.

Key Points:

The program defines a struct Node to represent individual nodes in the BST. Each node contains an integer data value, a left pointer, and a right pointer.

The BST operations are encapsulated in a BST class, which has private and public member functions.

Private member functions include:

createNode: Creates a new node with the given data and returns a pointer to it.

insertNode: Recursively inserts a new node with the provided data at the correct position in the BST, maintaining the BST property.

findMin: Finds the minimum value node (leftmost node) in the BST, used during deletion.

deleteNode: Recursively deletes the node with the given data from the BST while maintaining its property.

Public member functions include:

insert: Allows the user to insert a new node with the specified data into the BST.

remove: Allows the user to remove a node with the specified data from the BST.

display: Recursively displays the elements of the BST in ascending order.

displayTree: Displays the entire BST to the user.

The main function initializes the BST and provides a menu-driven interface to interact with the BST.

The user can choose from the following options:

Option 1: Insert a new node with a specified data value into the BST.

Option 2: Delete a node with a specified data value from the BST.

Option 3: Display the elements of the BST in ascending order.

Option 4: Exit the program.

The program handles invalid input by informing the user to try again.

The program continuously runs the menu until the user chooses to exit.

**CODE:**

#include <stdio.h>

#include <stdlib.h>

// Structure for a node in the BST

struct Node {

int data;

struct Node\* left;

struct Node\* right;

};

// Function to create a new node

struct Node\* createNode(int val) {

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

newNode->data = val;

newNode->left = NULL;

newNode->right = NULL;

return newNode;

}

// Function to insert a new node in the BST

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

if (root == NULL) {

return createNode(val);

}

if (val < root->data) {

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

} else {

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

}

return root;

}

// Function to find the node with the minimum value in a BST

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

while (root->left != NULL) {

root = root->left;

}

return root;

}

// Function to delete a node from the BST

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

if (root == NULL) {

return root;

}

if (val < root->data) {

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

} else if (val > root->data) {

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

} else {

// Node with only one child or no child

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;

}

// Node with two children: Get the inorder successor (smallest in the right subtree)

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

// Copy the inorder successor's data to this node

root->data = temp->data;

// Delete the inorder successor

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

}

return root;

}

// Function to perform inorder traversal

void inorderTraversal(struct Node\* root) {

if (root == NULL) {

return;

}

inorderTraversal(root->left);

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

inorderTraversal(root->right);

}

int main() {

struct Node\* root = NULL;

int choice, value;

do {

printf("1. Insert element\n");

printf("2. Delete element\n");

printf("3. Inorder traversal\n");

printf("4. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

printf("Enter value to insert: ");

scanf("%d", &value);

root = insert(root, value);

break;

case 2:

printf("Enter value to delete: ");

scanf("%d", &value);

root = deleteNode(root, value);

break;

case 3:

printf("Inorder traversal: ");

inorderTraversal(root);

printf("\n");

break;

case 4:

printf("Exiting...\n");

break;

default:

printf("Invalid choice\n");

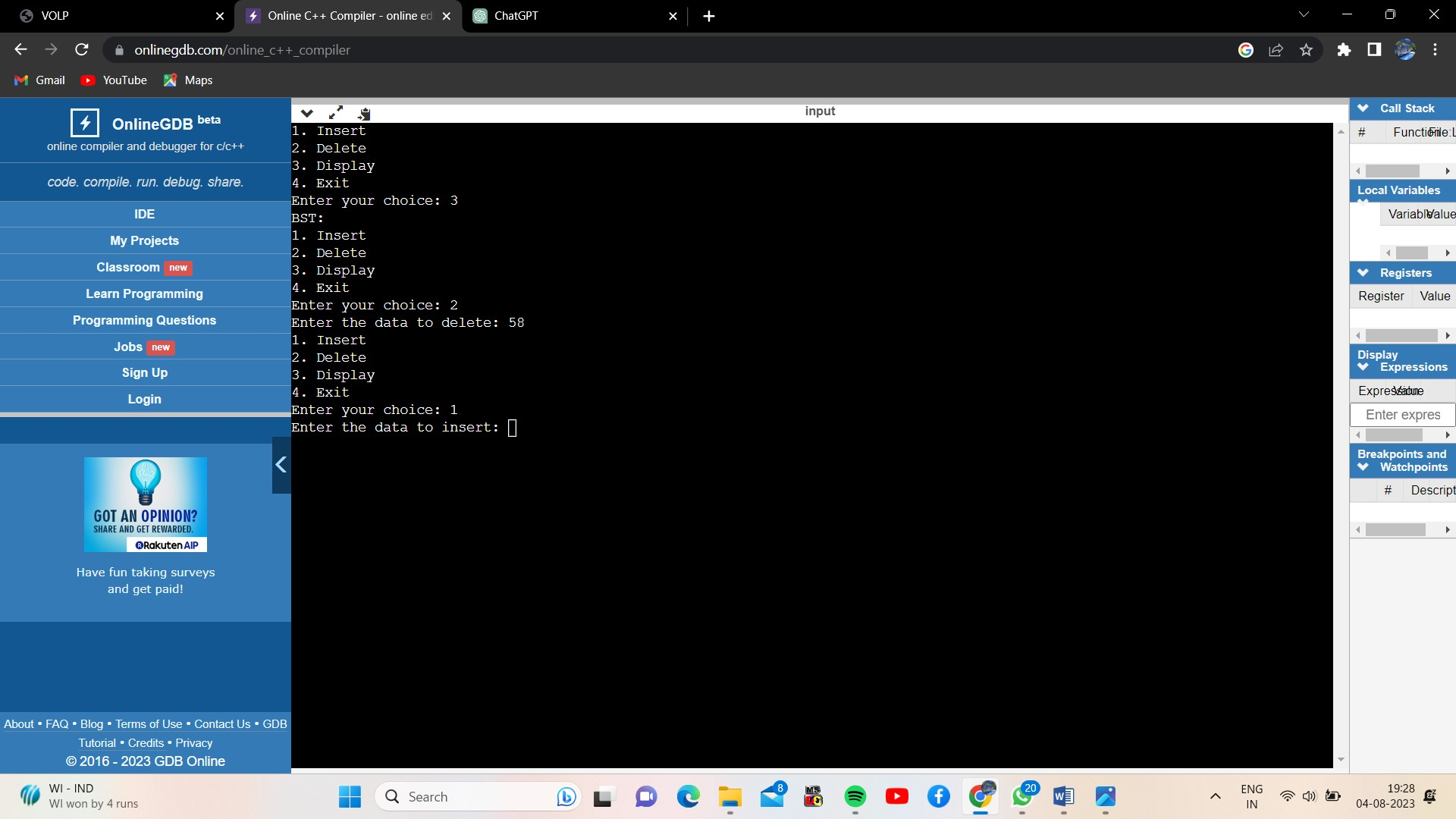
}

} while (choice != 4);

return 0;

}

**OUTPUT:-**



**CONCLUSION:-**

The provided C++ program implements a Binary Search Tree (BST) with insertion and deletion operations. The program utilizes a user-friendly menu to interact with the BST and allows users to insert nodes, delete nodes, and view the BST's contents in ascending order. The BST maintains its property, and the deletion function handles various cases. The program provides a simple and efficient way to manage data using a binary search tree structure.