Task # 1  
Write a program to insert 10 in the following BST

#include <iostream>

using namespace std;

class node{

    public:

        node \*left;

        node \*right;

        int data;

};

    node\* create\_node(int passed\_data){

        node \*newnode = new node;

        newnode->data = passed\_data;

        return newnode;

    }

    node\* insert\_in\_binary\_search\_tree(node\* root,int passed\_data){

        if(root==NULL){

            root = create\_node(passed\_data);

            root->left = NULL;

            root->right = NULL;

            return root;

        }

        else{

            if(passed\_data<root->data){

                root->left = insert\_in\_binary\_search\_tree(root->left, passed\_data);

            }

            else if(passed\_data>root->data){

                root->right = insert\_in\_binary\_search\_tree(root->right, passed\_data);

            }

            return root;

        }

    }

    void display(node\*root){

        if(root==NULL){

            cout << "";

        }

        else{

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

            display(root->left);

            display(root->right);

        }

    }

int main(){

    cout << "Binary Search Tree" << endl;

    node \*root = new node();

    root->data = 7;

    root->left = NULL;

    root->right = NULL;

    int opt;

    while(1){

        cout << "Enter 1 to insert in binary search tree" << endl;

        cout << "Enter 2 to display" << endl;

        cin >> opt;

        cout << endl;

        switch(opt){

            case 1:

            {

            int data;

            cout << "Enter the data : ";

            cin >> data;

            cout << endl;

            insert\_in\_binary\_search\_tree(root, data);

            break;

            }

            case 2:

            {

            display(root);

            cout << endl;

            break;

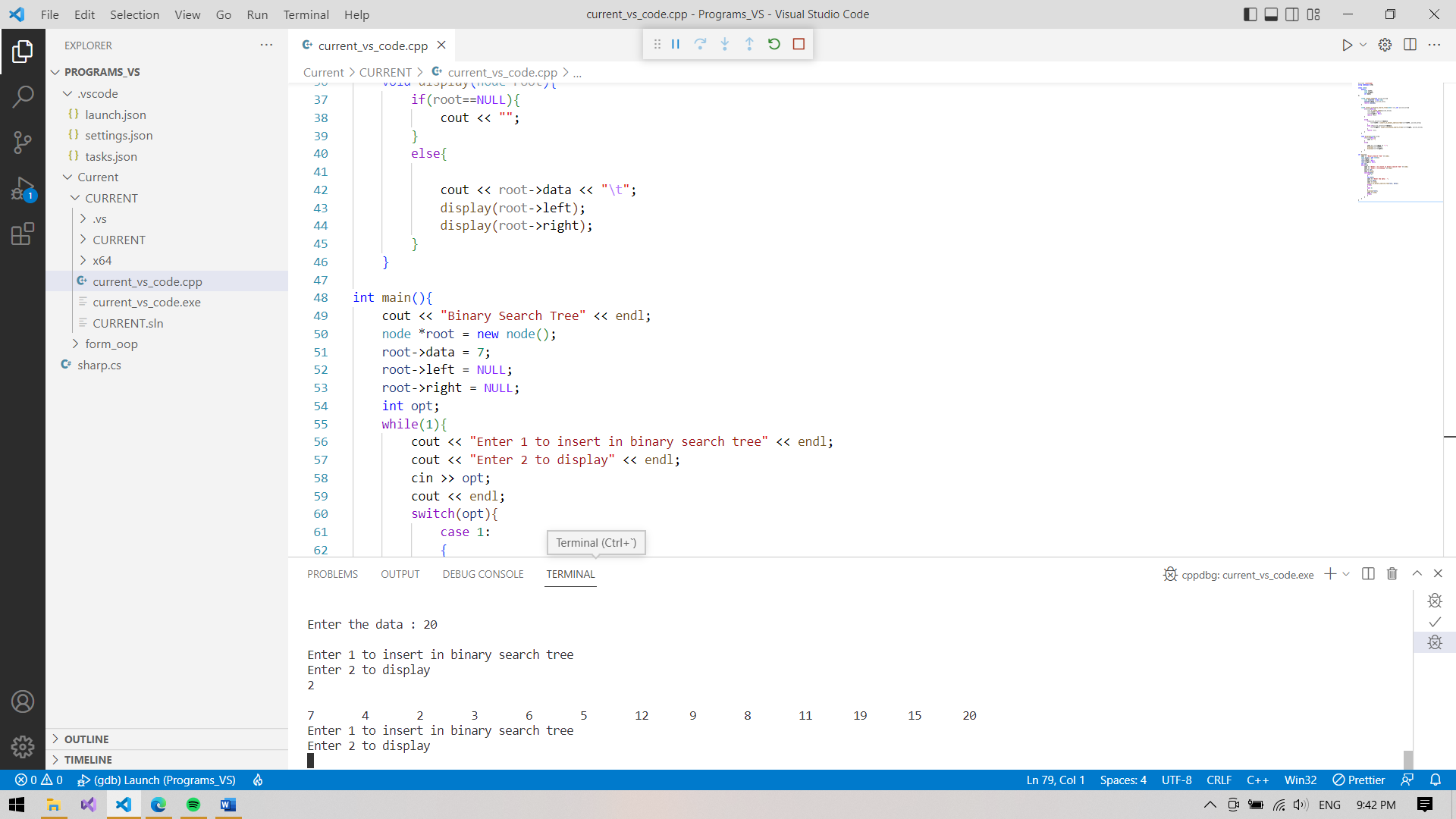
            }

        }

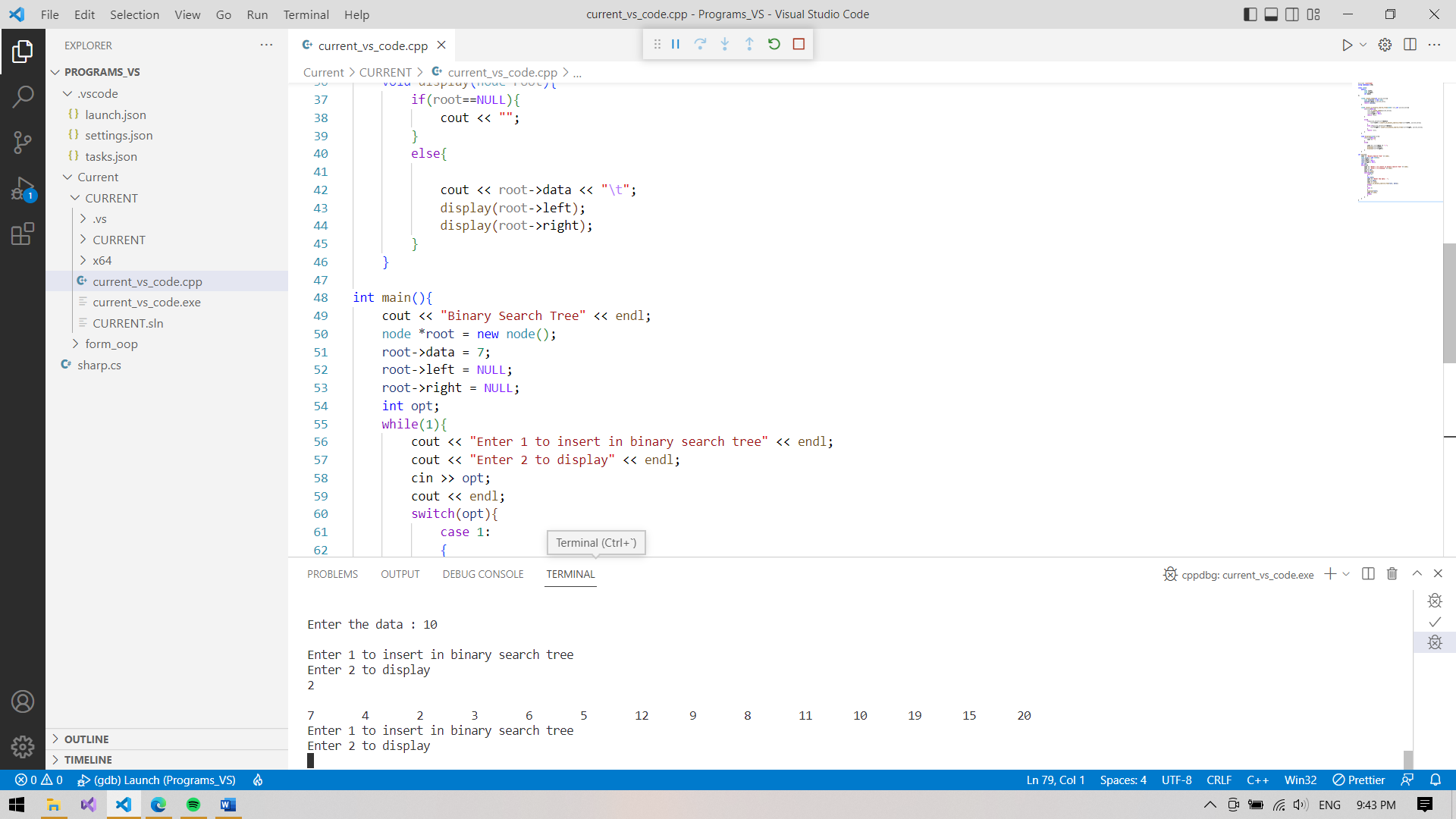
    }

}

**Original Tree:**



**After insertion of 10:**



Task # 2  
Write a program to delete 19 in the following BST

#include <iostream>

using namespace std;

class node {

public:

    node\* left;

    node\* right;

    int data;

};

node\* create\_node(int passed\_data) {

    node\* newnode = new node;

    newnode->data = passed\_data;

    return newnode;

}

node\* insert\_in\_binary\_search\_tree(node\* root, int passed\_data) {

    if (root == NULL) {

        root = create\_node(passed\_data);

        root->left = NULL;

        root->right = NULL;

        return root;

    }

    else {

        if (passed\_data < root->data) {

            root->left = insert\_in\_binary\_search\_tree(root->left, passed\_data);

        }

        else if (passed\_data > root->data) {

            root->right = insert\_in\_binary\_search\_tree(root->right, passed\_data);

        }

        return root;

    }

}

node\* delete\_node(node\* root, int data\_passed) {

    if (root->data == data\_passed) {

        if(root->left!=NULL){

            node \*temp = root->right;

            root = root->left;

            root->right = temp;

            return root;

        }

        else if(root->left==NULL&&root->right==NULL){

        root = NULL;

        return root;

        }

        else if(root->left==NULL){

        root = root->right;

        return root;

        }

        else if(root->right == NULL){

        root = root->left;

        return root;

        }

    }

    else {

        if (data\_passed < root->data) {

            root->left = delete\_node(root->left, data\_passed);

        }

        else if (data\_passed > root->data) {

            root->right = delete\_node(root->right, data\_passed);

        }

        return root;

    }

}

void display(node\* root) {

    if (root == NULL) {

        cout << "";

    }

    else {

        cout << root->data << "\t";

        display(root->left);

        display(root->right);

    }

}

int main() {

    cout << "Binary Search Tree" << endl;

    node\* root = new node();

    root->data = 7;

    root->left = NULL;

    root->right = NULL;

    int opt;

    while (1) {

        cout << "Enter 1 to insert in binary search tree" << endl;

        cout << "Enter 2 to display" << endl;

        cout << "Enter 3 to delete 19 node" << endl;

        cin >> opt;

        cout << endl;

        switch (opt) {

        case 1:

        {

            int data;

            cout << "Enter the data : ";

            cin >> data;

            cout << endl;

            insert\_in\_binary\_search\_tree(root, data);

            break;

        }

        case 2:

        {

            display(root);

            cout << endl;

            break;

        }

        case 3:

        {

            delete\_node(root, 19);

            break;

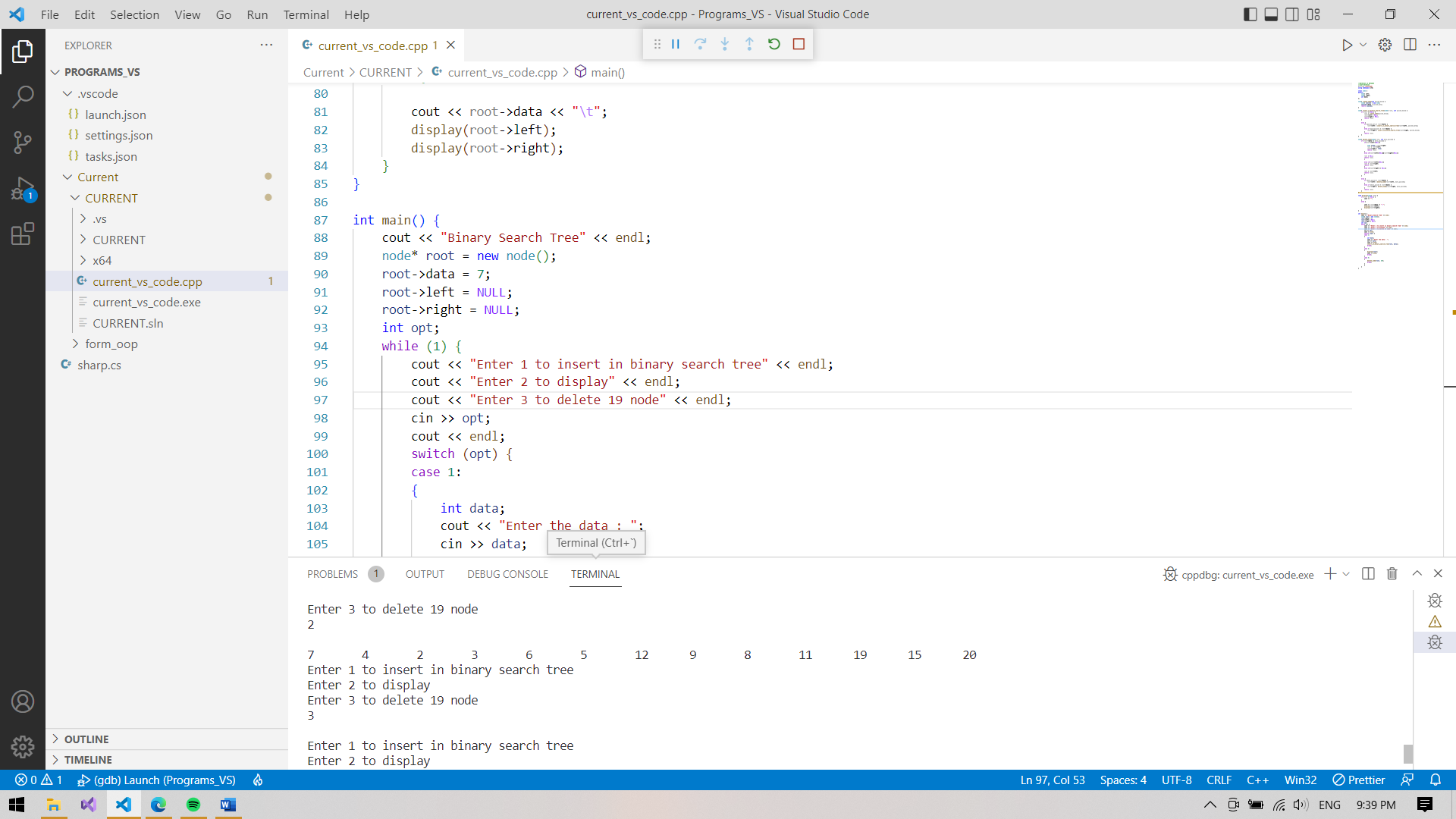
        }

        }

    }

}

**Original Tree:**



**After deletion of 19:**

