

**DSA**

**SUBMITTED BY**: IHSAN ULLAH KHAN

**ROLL NO:** 2330-0138

**CLASS SECTION**: FA23 / BS (AI)

 “On my honor, as student of Sir Syed CASE Institute Islamabad, I have neither given nor received unauthorized assistance on this academic work.”

**FINAL LAB**

**QUESTION 1 and 2:**

Question No.1 CLO-1

Write a program which should implement a Binary Search Tree. Elements of BST are objects of “student” class. “student” class contains attributes (privately defined) reg\_no(int), st\_name (string) and cgpa (float). “student” class should also contain member functions (publically defined); constructor, input and output functions. User will insert objects of class “student” in BST, values of attributes of objects will be provided by user.

Program should display the objects of student class in pre order, post order and in order traversal formats.

.....................................

.....................................

Question No.2 CLO-1

Based on question 1 Program should also search an object from BST  which is provided by user as input.

**CODE:**

#include <iostream>

#include <string>

using namespace std;

class Student {

private:

    int reg\_no;

    string st\_name;

    float cgpa;

public:

    Student() {

        reg\_no = 0;

        st\_name = "";

        cgpa = 0.0;

    }

    Student(int reg, string name, float grade) {

        reg\_no = reg;

        st\_name = name;

        cgpa = grade;

    }

    void input() {

        cout << "Enter Registration Number: ";

        cin >> reg\_no;

        cout << "Enter Student Name: ";

        cin.ignore();

        getline(cin, st\_name);

        cout << "Enter CGPA: ";

        cin >> cgpa;

    }

    void output() const {

        cout << "Registration Number: " << reg\_no << ", Name: " << st\_name << ", CGPA: " << cgpa << endl;

    }

    int getRegNo() const {

        return reg\_no;

    }

};

class BSTNode {

public:

    Student data;

    BSTNode\* left;

    BSTNode\* right;

    BSTNode(Student s) : data(s), left(nullptr), right(nullptr) {}

};

class BST {

private:

    BSTNode\* root;

    void insert(BSTNode\*& node, Student s) {

        if (node == nullptr) {

            node = new BSTNode(s);

        } else if (s.getRegNo() < node->data.getRegNo()) {

            insert(node->left, s);

        } else {

            insert(node->right, s);

        }

    }

    void preOrder(BSTNode\* node) const {

        if (node != nullptr) {

            node->data.output();

            preOrder(node->left);

            preOrder(node->right);

        }

    }

    void inOrder(BSTNode\* node) const {

        if (node != nullptr) {

            inOrder(node->left);

            node->data.output();

            inOrder(node->right);

        }

    }

    void postOrder(BSTNode\* node) const {

        if (node != nullptr) {

            postOrder(node->left);

            postOrder(node->right);

            node->data.output();

        }

    }

public:

    BST() : root(nullptr) {}

    void insert(Student s) {

        insert(root, s);

    }

    void displayPreOrder() const {

        preOrder(root);

    }

    void displayInOrder() const {

        inOrder(root);

    }

    void displayPostOrder() const {

        postOrder(root);

    }

    bool search(BSTNode\* node, int reg\_no, Student& result) const {

        if (node == nullptr) {

            return false;

        } else if (reg\_no == node->data.getRegNo()) {

            result = node->data;

            return true;

        } else if (reg\_no < node->data.getRegNo()) {

            return search(node->left, reg\_no, result);

        } else {

            return search(node->right, reg\_no, result);

        }

    }

public:

    bool search(int reg\_no, Student& result) const {

        return search(root, reg\_no, result);

    }

};

int main() {

    BST tree;

    int choice;

    do {

        cout << "1. Insert Student\n2. Display PreOrder\n3. Display InOrder\n4. Display PostOrder\n5. Search Student\n6. Exit\nEnter your choice: ";

        cin >> choice;

        switch (choice) {

            case 1: {

                Student s;

                s.input();

                tree.insert(s);

                break;

            }

            case 2:

                tree.displayPreOrder();

                break;

            case 3:

                tree.displayInOrder();

                break;

            case 4:

                tree.displayPostOrder();

                break;

            case 5: {

                int reg\_no;

                cout << "Enter Registration Number to search: ";

                cin >> reg\_no;

                Student result;

                if (tree.search(reg\_no, result)) {

                    cout << "Student found: ";

                    result.output();

                } else {

                    cout << "Student not found." << endl;

                }

                break;

            }

            case 6:

                cout << "Exiting..." << endl;

                break;

            default:

                cout << "Invalid choice. Please try again." << endl;

        }

    } while (choice != 6);

    return 0;

}

**OUTPUT:**

1. Insert Student

2. Display PreOrder

3. Display InOrder

4. Display PostOrder

5. Search Student

6. Exit

Enter your choice: 1

Enter Registration Number: 1

Enter Student Name: sd

Enter CGPA: 3

1. Insert Student

2. Display PreOrder

3. Display InOrder

4. Display PostOrder

5. Search Student

6. Exit

Enter your choice: 1

Enter Registration Number: 2

Enter Student Name: wdd

Enter CGPA: 3

1. Insert Student

2. Display PreOrder

3. Display InOrder

4. Display PostOrder

5. Search Student

6. Exit

Enter your choice: 1

Enter Registration Number: 3

Enter Student Name: ewd

Enter CGPA: 2

1. Insert Student

2. Display PreOrder

3. Display InOrder

4. Display PostOrder

5. Search Student

6. Exit

Enter your choice: 5

Enter Registration Number to search: 2

Student found: Registration Number: 2, Name: wdd, CGPA: 3

1. Insert Student

2. Display PreOrder

3. Display InOrder

4. Display PostOrder

5. Search Student

6. Exit

Enter your choice: 3

Registration Number: 1, Name: sd, CGPA: 3

Registration Number: 2, Name: wdd, CGPA: 3

Registration Number: 3, Name: ewd, CGPA: 2

1. Insert Student

2. Display PreOrder

3. Display InOrder

4. Display PostOrder

5. Search Student

6. Exit

Enter your choice: 4

Registration Number: 3, Name: ewd, CGPA: 2

Registration Number: 2, Name: wdd, CGPA: 3

Registration Number: 1, Name: sd, CGPA: 3

1. Insert Student

2. Display PreOrder

3. Display InOrder

4. Display PostOrder

5. Search Student

6. Exit

Enter your choice: 2

Registration Number: 1, Name: sd, CGPA: 3

Registration Number: 2, Name: wdd, CGPA: 3

Registration Number: 3, Name: ewd, CGPA: 2

1. Insert Student

2. Display PreOrder

3. Display InOrder

4. Display PostOrder

5. Search Student

6. Exit

…………………………………………………………………….

**QUESTION 2:**

**CODE:**

#include <iostream>

#include <string>

using namespace std;

class Node

{

public:

    string title;

    string author;

    int price;

    Node \*next;

    Node(string title, string author, int price)

    {

        this->title = title;

        this->author = author;

        this->price = price;

        next = nullptr;

    }

    ~Node()

    {

    }

};

class Stack

{

private:

    Node \*top;

public:

    Stack()

    {

        top = nullptr;

    }

    void push(string title, string author, int price)

    {

        Node \*newNode = new Node(title, author, price);

        newNode->next = top;

        top = newNode;

    }

    void search(string title)

    {

        Node \*current = top;

        while (current != nullptr)

        {

            if (current->title == title)

            {

                cout << "Book found: " << current->title << ", " << current->author << ", $" << current->price << endl;

                return;

            }

            current = current->next;

        }

        cout << "Book not found." << endl;

    }

    void deleteBook(string title)

    {

        if (top == nullptr)

        {

            cout << "Stack is empty. Cannot delete." << endl;

            return;

        }

        if (top->title == title)

        {

            Node \*temp = top;

            top = top->next;

            delete temp;

            cout << "Book deleted from stack." << endl;

            return;

        }

        Node \*current = top;

        Node \*previous = nullptr;

        while (current != nullptr)

        {

            if (current->title == title)

            {

                previous->next = current->next;

                delete current;

                cout << "Book deleted from stack." << endl;

                return;

            }

            previous = current;

            current = current->next;

        }

        cout << "Book not found." << endl;

    }

    void display()

    {

        if (top == nullptr)

        {

            cout << "Stack is empty." << endl;

            return;

        }

        Node \*current = top;

        cout << "Books in stack:" << endl;

        while (current != nullptr)

        {

            cout << "Title: " << current->title

                 << ", Author: " << current->author

                 << ", Price: $" << current->price << endl;

            current = current->next;

        }

    }

};

int main()

{

    Stack bookStack;

    bookStack.push("lahor", "nawaz", 10);

    bookStack.push("911", "Osama", 15);

    bookStack.push("Pakhtoon", "Bacha khan", 158);

    bookStack.push("Pakistan", "quaideazam", 12);

    bookStack.search("911");

    bookStack.deleteBook("Pakistan");

    bookStack.display();

    return 0;

}

**OUTPUT:**

Book found: 911, Osama, $15

Book deleted from stack.

Books in stack:

Title: Pakhtoon, Author: Bacha khan, Price: $158

Title: 911, Author: Osama, Price: $15

Title: lahor, Author: nawaz, Price: $10

…………………………………………………………………….