DR. D. Y. PATIL SCHOOL OF SCIENCE & TECHNOLOGY
DR. D. Y. PATIL VIDYAPEETH, PUNE
(Deemed to be University)
(Accredited (3rd cycle) by NAAC with a CGPA of 3.64 on four-point scale at 'A++' Grade)
(Declared as Category - I University by UGC Under Graded Autonomy Regulations, 2018)
(An ISO 9001: 2015 and 14001:2015 Certified University and Green Education Campus)

Date: 11/3/24

Assignment No: 8

Problem Statement:

(a) Design, Develop and Implement a menu driven Program in for the following operations on Binary

Search Tree(BST) of Integers

(b) Create a BST of N Integers: 6, 9, 5, 2, 8, 15, 24, 14, 7, 8, 5, 2

(c) Traverse the BST in Inorder, Preorder and Post Order

(d) Search the BST for a given element (KEY) and report the appropriate message

Algorithm:

1. Define a structure Node to represent a node in the Binary Search Tree (BST) where each node
contains: An integer data to store the value. Pointers left and right to point to the left and right child
nodes, respectively.
2. Define a class 'BinarySearchTree' to encapsulate BST operations. Declare a pointer root to
represent the root of the BST.
3. Implement the 'insert' function to insert a new node with a given value into the BST.
- Start from the root and recursively traverse the tree.
- If the tree is empty, create a new node and make it the root.
- If the value is less than the current node's value, move to the left subtree.
- If the value is greater than the current node's value, move to the right subtree.
- Repeat the process until finding a suitable position to insert the new node.
4. Implement three traversal functions: 'inorderTraversal','preorderTraversal', and 'postorderTraversal'.
- Each traversal function recursively visits all nodes in the BST.
- In inorder traversal, visit the left subtree, then the root, then the right subtree.
- In preorder traversal, visit the root, then the left subtree, then the right subtree.

- In postorder traversal, visit the left subtree, then the right subtree, then the root.
5. Implement the ‘search‘ function to search for a given key in the BST.
- Start from the root and recursively traverse the tree.
- If the key matches the current node's value, return true.
- If the key is less than the current node's value, search in the left subtree.
- If the key is greater than the current node's value, search in the right subtree.
- Repeat the process until finding the key or reaching a null pointer
6. Create an object of ‘BinarySearchTree‘.
- Insert the provided elements into the BST.
- Call the traversal functions to print the elements in inorder, preorder, and postorder.
- Perform a search operation for a given key and print the result.

Source Code:

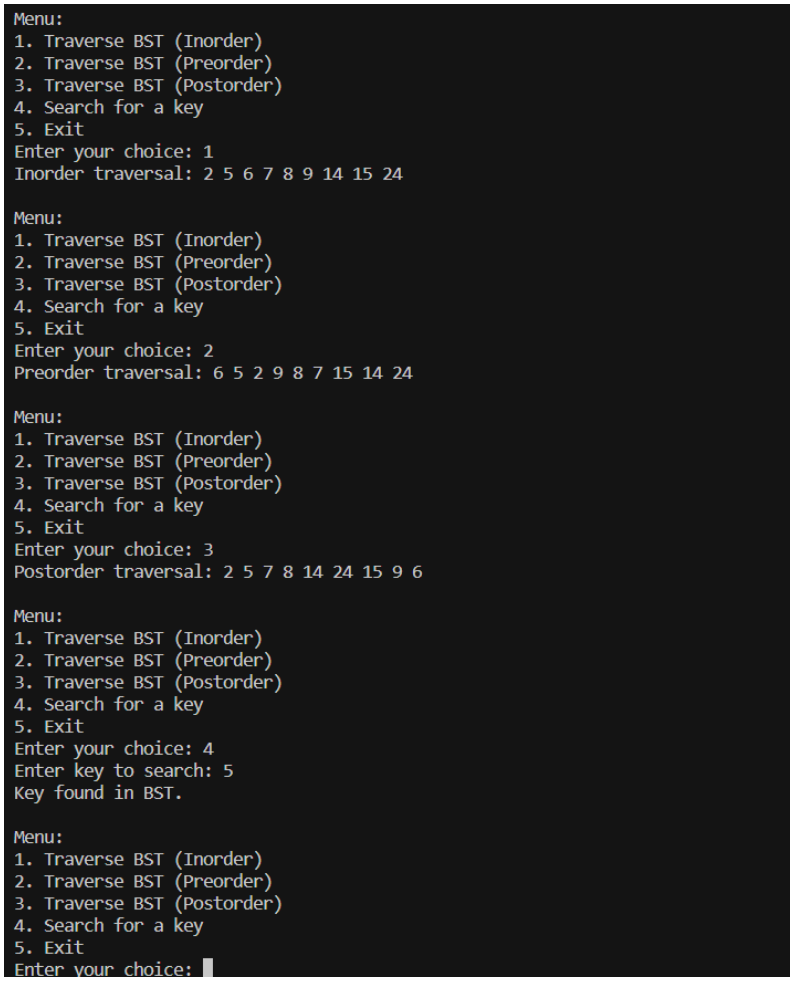
```
#include <stdio.h>
#include <stdlib.h>

// Define the structure of 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 value) {
    struct node *newNode = (struct node *)malloc(sizeof(struct node));
    newNode->data = value;
    newNode->left = newNode->right = NULL;
    return newNode;
}

// Function to insert a new node into BST
struct node *insert(struct node *root, int value) {
    if (root == NULL) {
        return createNode(value);
    }
    if (value < root->data) {
        root->left = insert(root->left, value);
    } else if (value > root->data) {
        root->right = insert(root->right, value);
    }
    return root;
}
```

```
// Function to perform inorder traversal of BST
void inorder(struct node *root) {
    if (root != NULL) {
        inorder(root->left);
        printf("%d ", root->data);
        inorder(root->right);
    }
}

// Function to perform preorder traversal of BST
void preorder(struct node *root) {
    if (root != NULL) {
        printf("%d ", root->data);
        preorder(root->left);
        preorder(root->right);
    }
}

// Function to perform postorder traversal of BST
void postorder(struct node *root) {
    if (root != NULL) {
        postorder(root->left);
        postorder(root->right);
        printf("%d ", root->data);
    }
}

// Function to search for a key in BST
struct node *search(struct node *root, int key) {
    if (root == NULL || root->data == key) {
        return root;
    }
    if (root->data < key) {
        return search(root->right, key);
    }
    return search(root->left, key);
}

int main() {
    struct node *root = NULL;
    int choice, key;

    // Creating the BST with given elements
    int elements[] = {6, 9, 5, 2, 8, 15, 24, 14, 7, 8, 5, 2};
```

```
int numElements = sizeof(elements) / sizeof(elements[0]);
for (int i = 0; i < numElements; i++) {
    root = insert(root, elements[i]);
}

do {
    printf("\nMenu:\n");
    printf("1. Traverse BST (Inorder)\n");
    printf("2. Traverse BST (Preorder)\n");
    printf("3. Traverse BST (Postorder)\n");
    printf("4. Search for a key\n");
    printf("5. Exit\n");
    printf("Enter your choice: ");
    scanf("%d", &choice);

    switch (choice) {
        case 1:
            printf("Inorder traversal: ");
            inorder(root);
            printf("\n");
            break;
        case 2:
            printf("Preorder traversal: ");
            preorder(root);
            printf("\n");
            break;
        case 3:
            printf("Postorder traversal: ");
            postorder(root);
            printf("\n");
            break;
        case 4:
            printf("Enter key to search: ");
            scanf("%d", &key);
            if (search(root, key) != NULL) {
                printf("Key found in BST.\n");
            } else {
                printf("Key not found in BST.\n");
            }
            break;
        case 5:
            printf("Exiting program.\n");
            break;
        default:
            printf("Invalid choice. Please enter a number between 1 and 5.\n");
```

```
        }
    } while (choice != 5);

    return 0;
}
```

Sample Output:



GITHUB: -Advanced-Data-Structure/ at main · TejalNehete/-Advanced-Data-Structure (github.com)