Program 10 : Binary Search Tree

Develop a menu driven Program in C for the following operations on Binary Search Tree (BST) of Integers .

a. Create a BST of N Integers:

6, 9, 5, 2, 8, 15, 24, 14, 7, 8, 5, 2

b. Traverse the BST in Inorder, Preorder and Post Order

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

d. Exit

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

// Define a structure for BST nodes
struct BSTNode {
   int data;
   struct BSTNode *left, *right;
};

// Function to create a new BST node
struct BSTNode* newNode(int item) {
   struct BSTNode* temp = (struct BSTNode*)malloc(sizeof(struct BSTNode));
   temp->data = item;
   temp->left = temp->right = NULL;
   return temp;
}

// Function to insert a new node with a given key in the BST
struct BSTNode* insert(struct BSTNode* node, int key) {
   // If the tree is empty, return a new node
   if (node == NULL) return newNode(key);

   // Otherwise, recur down the tree
   if (key < node->data)
      node->left = insert(node->left, key);
```

```
    else if (key > node->data)
        node->right = insert(node->right, key);

    // return the (unchanged) node pointer
    return node;
}

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

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

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

// Function to search a given key in a given BST
struct BSTNode* search(struct BSTNode* root, int key) {
    // Base Cases: root is null or key is present at root
    if (root == NULL || root->data == key)
      return root;
```

```
    // Key is greater than root's key
    if (root->data < key)
      return search(root->right, key);

    // Key is smaller than root's key
    return search(root->left, key);
}

// Driver Program to test above functions
int main() {
    struct BSTNode* root = NULL;
    int choice, key,size=100;
    int data[size],n;
    printf("Enter No. of elements in BST\n");
    scanf("%d",&n);
    printf("Enter Elements into BST:\n");
    for(int i=0;i<n;i++)
          scanf("%d",&data[i]);
          // Creating BST
    for(int i = 0; i < n; i++) {
      root = insert(root, data[i]);
    }

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

      switch(choice) {
        case 1:
          printf("Inorder traversal: ");
          inorder(root);
          break;
        case 2:
          printf("Preorder traversal: ");
```

```
            preorder(root);
            break;
        case 3:
            printf("Postorder traversal: ");
            postorder(root);
            break;
        case 4:
            printf("Enter the element to search: ");
            scanf("%d", &key);
            if (search(root, key) != NULL)
                printf("Element found\n");
            else
                printf("Element not found\n");
            break;
        case 5:
            printf("Exiting...\n");
            break;
        default:
            printf("Invalid choice\n");
    }
  } while (choice != 5);

  return 0;
}
```