Practice Problem 7

Data Structure Lab ID:2023000000033

1. Write a code to implement the insertion of the following values into a Binary Search Tree (BST):

35, S0, 40, 25, 30, 60, 78, 20, 28.

The code should insert nodes in a way that maintains the BST properties.

2. Write a code to search for a value which is taken from a user in the BST. If the value is found, return True; otherwise, return False.

3. Write a code to find and return the minimum (lowest) value in the given BST.

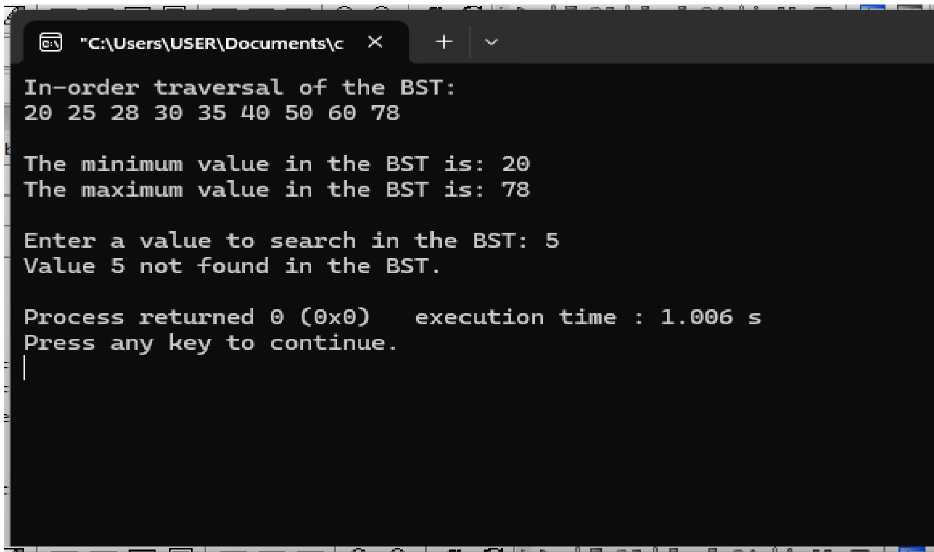
4.Write a code to find and return the maximum (highest) value in the given BST.

<u>Solution:</u>

```
#include <stdio.h>
#include <stdlib.h>
// Define the structure for a tree node
struct Node {
   int data;
   struct Node* left;
   struct Node* right;
};

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

// Function for in-order traversal
void inorderTraversal(struct Node* root)
{
   if (root == NULL) return;
   inorderTraversal(root->left);
   printf("%d ", root->data);
   inorderTraversal(root->right);
}
```



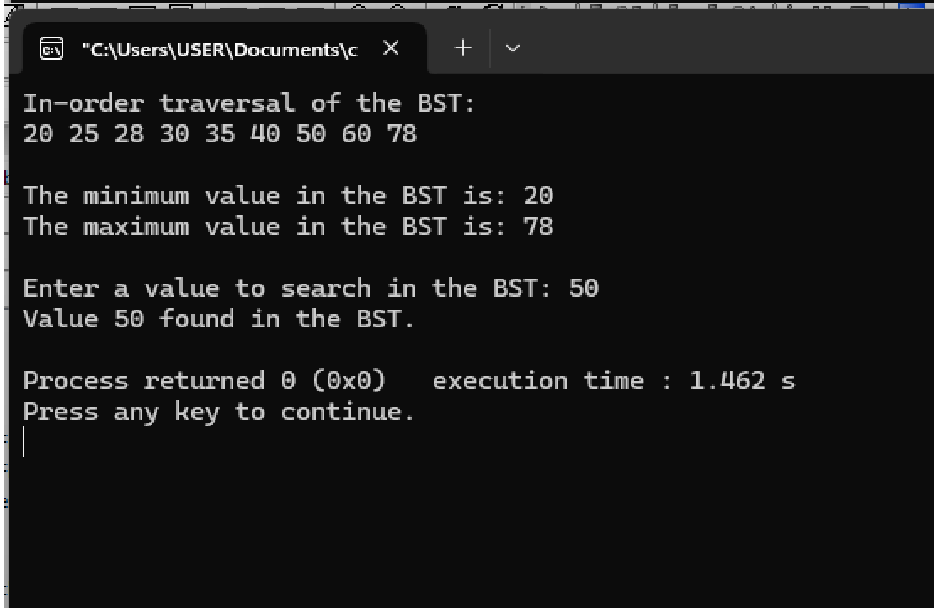
"C:\Users\USER\Documents\c

```
In-order traversal of the BST:
20 25 28 30 35 40 50 60 78

The minimum value in the BST is: 20
The maximum value in the BST is: 78

Enter a value to search in the BST: 5
Value 5 not found in the BST.

Process returned 0 (0x0)   execution time : 1.006 s
Press any key to continue.
```



"C:\Users\USER\Documents\c

```
In-order traversal of the BST:
20 25 28 30 35 40 50 60 78

The minimum value in the BST is: 20
The maximum value in the BST is: 78

Enter a value to search in the BST: 50
Value 50 found in the BST.

Process returned 0 (0x0)   execution time : 1.462 s
Press any key to continue.
```

```
// Function to insert a new node in the binary tree------------1 Ans
struct Node* insert(struct Node* root, int data) {
   if (root == NULL) {
      return createNode(data);
   }
   if (data < root->data) {
      root->left = insert(root->left, data);
   } else {
      root->right = insert(root->right, data);
   }
   return root;
}

// Function to find the minimum value in the BST--------------------3 Ans
int findMin(struct Node* root) {
   struct Node* current = root;
   while (current && current->left != NULL) {
      current = current->left;
   }
   return current->data;
}

// Function to find the maximum value in the BST--------------------4 Ans
int findMax(struct Node* root) {
   struct Node* current = root;
   while (current && current->right != NULL) {
      current = current->right;
   }
   return current->data;
}

// Function to search for a value in the BST--------------2 Ans
int search(struct Node* root, int value) {
   if (root == NULL) {
      return 0; // Value not found
   }
   if (root->data == value) {
      return 1; // Value found
   }
   if (value < root->data) {
      return search(root->left, value);
   } else {
      return search(root->right, value);
```

```
    }
}

int main() {
    struct Node* root = NULL;
    int value;

    // Insert nodes into the binary tree
    root = insert(root, 35);
    insert(root, 50);
    insert(root, 40);
    insert(root, 25);
    insert(root, 30);
    insert(root, 60);
    insert(root, 78);
    insert(root, 20);
    insert(root, 28);

    // Perform in-order traversal------------------------------------print of 1
    printf("In-order traversal of the BST:\n");
    inorderTraversal(root);
    printf("\n");
    printf("\n");

    // Find and print the minimum and maximum values in the BST--------------------print of 3&4
    int minValue = findMin(root);
    int maxValue = findMax(root);
    printf("The minimum value in the BST is: %d\n", minValue);
    printf("The maximum value in the BST is: %d\n", maxValue);printf("\n");

    // Asking the user for a value to search
    printf("Enter a value to search in the BST: ");
    scanf("%d", &value);

    // Search for the value in the BST -----------------------------------------print of 2
    if (search(root, value)) {
        printf("Value %d found in the BST.\n", value);
    } else {
        printf("Value %d not found in the BST.\n", value);
    }
    return 0;
}
```