SOUTHEAST UNIVERSITY

Meeting the Challenges of Time

Department Of CSE

Cse Assignment

Assignment No:07

Date of submission:

Course name : CSE LAB

Course code : 241

Section : 9

Student's Name :Nazmul Hasan

Student's ID :2023100000130

Submitted To :Mr. Muhammed Yeaseen Morshed Abid

[lecturer, Department of CSE]

Initial : YMA

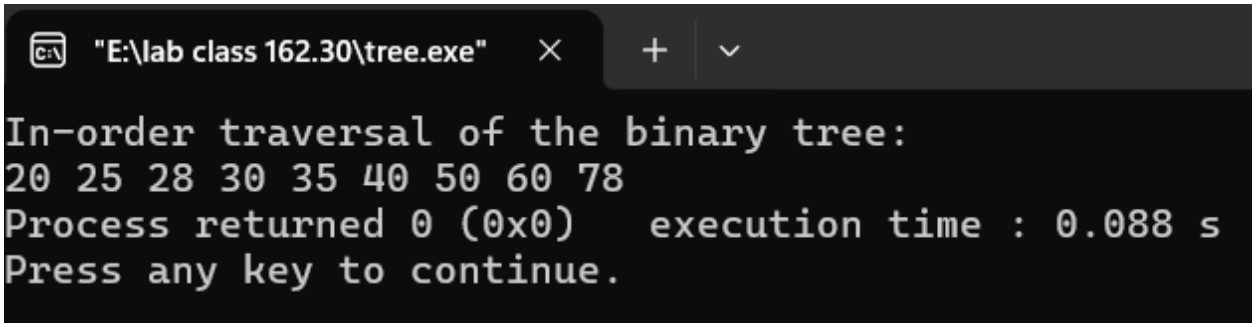
Code 1:

```
#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);
```

```
}

// Function to insert a new node in the binary tree
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;
}

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

    // 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
    printf("In-order traversal of the binary tree:\n");
    inorderTraversal(root);

    return 0;
}
```

Terminal:



Code 2:

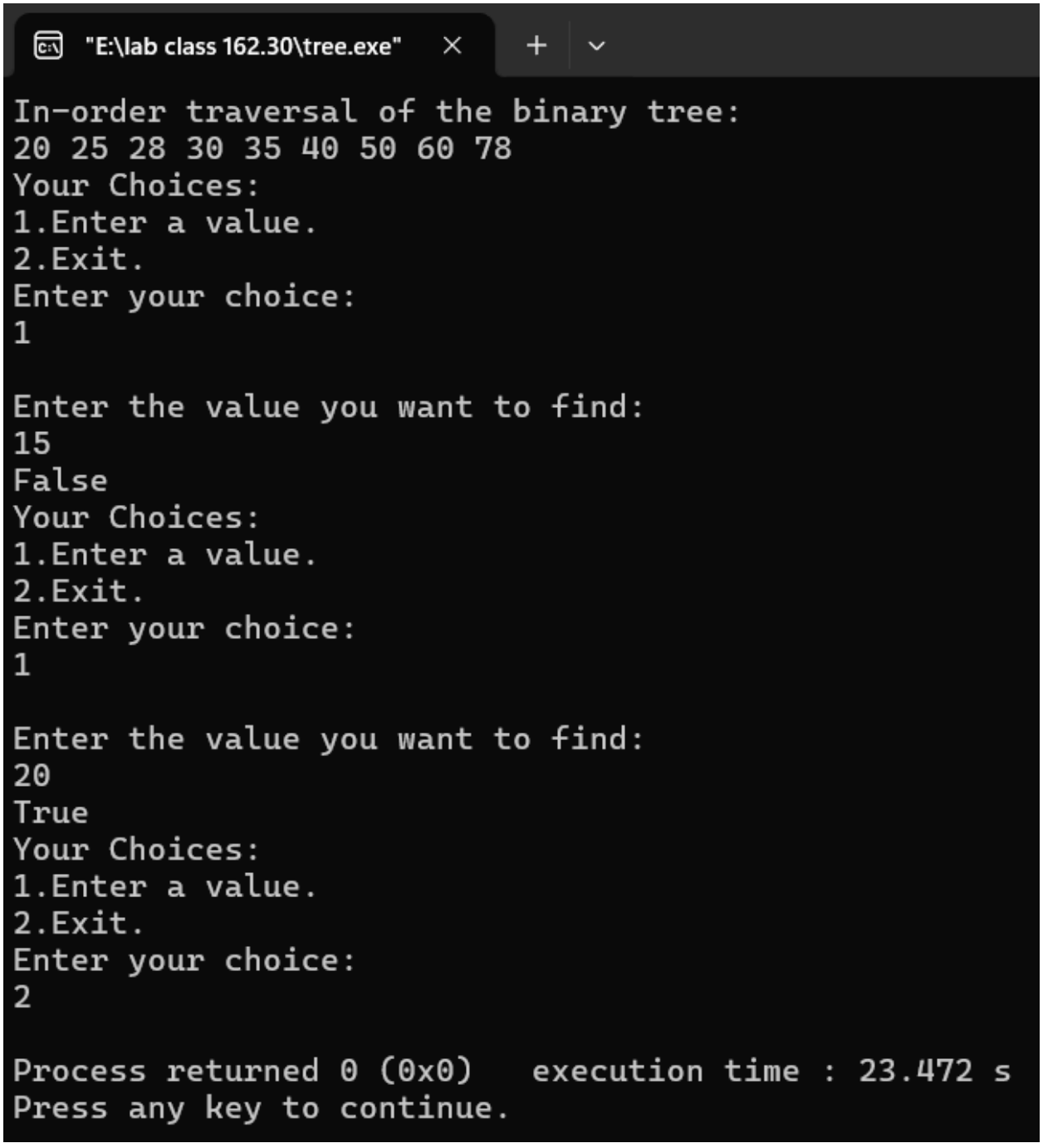
```
#include <stdio.h>
#include <stdlib.h>

// Define the structure for a tree node
typedef struct Node {
   int data;
   struct Node* left;
   struct Node* right;
}Node;

// Function to create a new node
   Node* createNode(int data) {
   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(Node* root) {
   if (root == NULL) return;

   inorderTraversal(root->left);
   printf("%d ", root->data);
   inorderTraversal(root->right);
```

```
}

// Function to insert a new node in the binary tree
   Node* insert(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 search for a value in the binary search tree
   Node* search(Node* root, int key) {
// Base case: root is NULL or key is present at root
   if (root == NULL) {
      return 0;
   }
   if(root->data == key)
   {
      return 1;
   }
// Key is greater than root's data
```

```
    if (key > root->data) {
        return search(root->right, key); // Search in the right subtree
}
// Key is smaller than root's data
    return search(root->left, key); // Search in the left subtree
}
int main() {
     Node* root = NULL;

    // 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
    printf("In-order traversal of the binary tree:\n");
    inorderTraversal(root);

    //finding value
    int choice = 0;
    while(1){
```

```
        int key;
        printf("\nYour Choices:\n");
        printf("1.Enter a value.\n");
        printf("2.Exit.\n");
        printf("Enter your choice: \n");
        scanf("%d", &choice);
        if(choice == 1)
        {
            printf("\nEnter the value you want to find:\n");
            scanf("%d", &key);
        }
        else if(choice == 2)
        {
            break;
        }
        search(root, key);
        if(search(root, key)){
        printf("True");
    }
    else{
        printf("False");
    }
    }
    return 0;
}
```

Terminal:


"E:\lab class 162.30\tree.exe"
In-order traversal of the binary tree:
20 25 28 30 35 40 50 60 78
Your Choices:
1.Enter a value.
2.Exit.
Enter your choice:
1
Enter the value you want to find:
15
False
Your Choices:
1.Enter a value.
2.Exit.
Enter your choice:
1
Enter the value you want to find:
20
True
Your Choices:
1.Enter a value.
2.Exit.
Enter your choice:
2
Process returned 0 (0x0) execution time : 23.472 s
Press any key to continue.

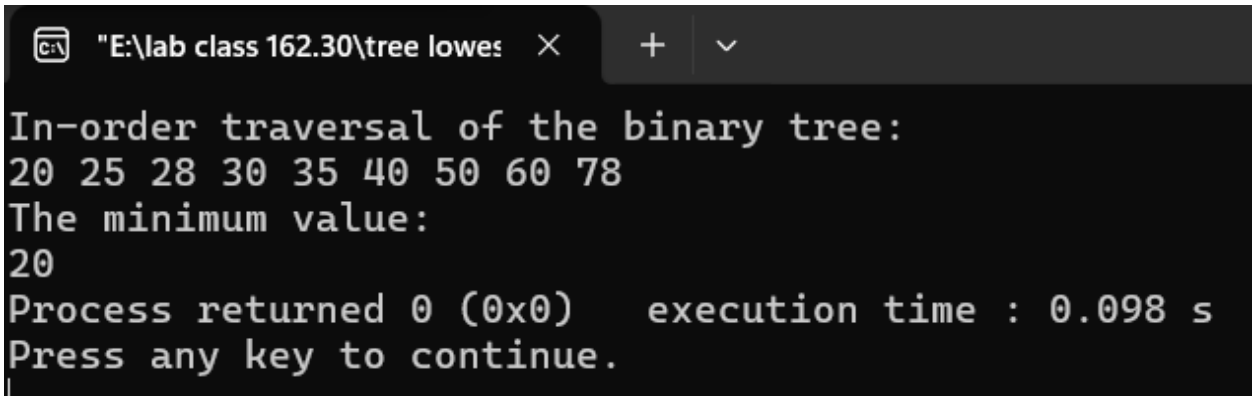
Code 3:

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

// Define the structure for a tree node
typedef struct Node {
   int data;
   struct Node* left;
   struct Node* right;
}Node;
// Function to create a new node
   Node* createNode(int data) {
   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(Node* root) {
   if (root == NULL) return;

   inorderTraversal(root->left);
   printf("%d ", root->data);
   inorderTraversal(root->right);
}
```

```
// Function to insert a new node in the binary tree
   Node* insert(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 lowest (minimum) value in the binary search tree
   int findMinValue(struct Node* root) {
   struct Node* current = root;
// Loop down to find the leftmost leaf (smallest value)
   while (current && current->left != NULL) {
   current = current->left;
}
   return current->data;
}
int main() {
    Node* root = NULL;

   // 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
    printf("In-order traversal of the binary tree:\n");
    inorderTraversal(root);

    printf("\nThe minimum value:\n");
    printf("%d",findMinValue(root) );

    return 0;
}
```

Terminal:



Code 4:

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

// Define the structure for a tree node
typedef struct Node {
   int data;
   struct Node* left;
   struct Node* right;
}Node;

// Function to create a new node
   Node* createNode(int data) {
   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(Node* root) {
   if (root == NULL) return;
   inorderTraversal(root->left);
   printf("%d ", root->data);
   inorderTraversal(root->right);
}
```

```
// Function to insert a new node in the binary tree
   Node* insert(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 highest (maximum) value in the binary search tree
   int findMaxValue(struct Node* root) {
   struct Node* current = root;
// Loop down to find the leftmost leaf (smallest value)
   while (current && current->right != NULL) {
   current = current->right;
}
   return current->data;
}

int main() {
    Node* root = NULL;

   // 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
    printf("In-order traversal of the binary tree:\n");
    inorderTraversal(root);

    printf("\nThe maximum value:\n");
    printf("%d",findMaxValue(root) );

    return 0;
}
```

Terminal: