**Reg. No. 12212256 Academic Tasks (22232)**

****

**Academic Task Number: 3**

**Course code: CAP770 Course title: Advanced Data Structures**

**Date of allotment: 12-March-2023 Date of submission: 12-March 2023**

**Max Marks: 50 Section: D2217/D2221-G2**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Question Number** | **Question Statement** | **Course Outcome** | **Bloom’s level** | **Marks per Question** |
| **Q2**  **(EVEN)** | **Implement a BST for given data and then search the node for the given value**  **[5,3,6,7,2,8,4]**  **Search node with value 2** | **CO2** | **L3: Apply** | **25** |
| **Viva** |  | **CO1,CO2** | **L1:Remeber**  **L2:**  **Understand** | **25** |

**Code**

**#include <stdio.h>**

**#include <stdlib.h>**

**struct node {**

**int data;**

**struct node \*left;**

**struct node \*right;**

**};**

**struct node\* newNode(int data) {**

**struct node\* node = (struct node\*)malloc(sizeof(struct node));**

**node->data = data;**

**node->left = NULL;**

**node->right = NULL;**

**return node;**

**};**

**struct node\* insert(struct node\* node, int data) {**

**if (node == NULL) {**

**return newNode(data);**

**}**

**if (data < node->data) {**

**node->left = insert(node->left, data);**

**} else if (data > node->data) {**

**node->right = insert(node->right, data);**

**}**

**return node;**

**}**

**struct node\* search(struct node\* root, int data) {**

**if (root == NULL || root->data == data) {**

**return root;**

**}**

**if (data < root->data) {**

**return search(root->left, data);**

**} else {**

**return search(root->right, data);**

**}**

**}**

**void inorder(struct node\* node) {**

**if (node != NULL) {**

**inorder(node->left);**

**printf("%d ", node->data);**

**inorder(node->right);**

**}**

**}**

**int main() {**

**struct node\* root = NULL;**

**int data[] = {5, 3, 6, 7, 2, 8, 4};**

**int n = sizeof(data) / sizeof(data[0]);**

**for (int i = 0; i < n; i++) {**

**root = insert(root, data[i]);**

**}**

**printf("Inorder traversal of the BST: ");**

**inorder(root);**

**printf("\n");**

**int searchValue = 2;**

**struct node\* result = search(root, searchValue);**

**if (result != NULL) {**

**printf("Node with value %d found in the BST\n", searchValue);**

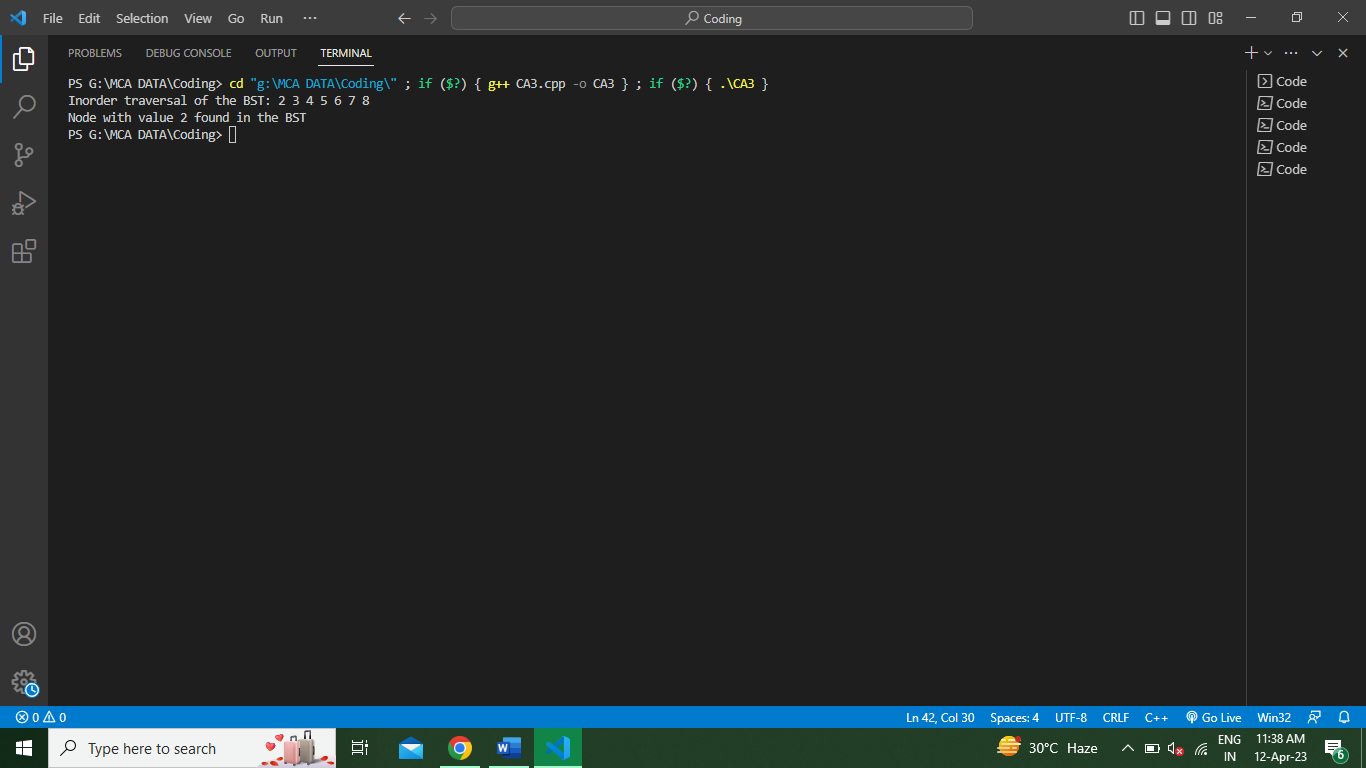
**} else {**

**printf("Node with value %d not found in the BST\n", searchValue);**

**}**

**return 0;**

**}**

**Screenshot of Output**