

# Rajalakshmi Engineering College

Name: Jayasree D  
Email: 241801100@rajalakshmi.edu.in  
Roll no:  
Phone: 9025821157  
Branch: REC  
Department: I AI & DS FB  
Batch: 2028  
Degree: B.E - AI & DS

Scan to verify results



## NeoColab\_REC\_CS23231\_DATA STRUCTURES

### REC\_DS using C\_Week 5\_COD\_Question 5

Attempt : 1  
Total Mark : 10  
Marks Obtained : 10

#### Section 1 : Coding

##### 1. Problem Statement

In his computer science class, John is learning about Binary Search Trees (BST). He wants to build a BST and find the maximum value in the tree.

Help him by writing a program to insert nodes into a BST and find the maximum value in the tree.

##### ***Input Format***

The first line of input consists of an integer N, representing the number of nodes in the BST.

The second line consists of N space-separated integers, representing the values of the nodes to insert into the BST.

##### ***Output Format***

The output prints the maximum value in the BST.

Refer to the sample output for formatting specifications.

### ***Sample Test Case***

Input: 5

10 5 15 2 7

Output: 15

### ***Answer***

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

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

```
struct TreeNode {  
    int data;  
    struct TreeNode* left;  
    struct TreeNode* right;  
};
```

```
struct TreeNode* createNode(int key) {  
    struct TreeNode* newNode = (struct TreeNode*)malloc(sizeof(struct  
TreeNode));  
    newNode->data = key;  
    newNode->left = newNode->right = NULL;  
    return newNode;  
}
```

```
// You are using GCC
```

```
#include <iostream>
```

```
using namespace std;
```

```
class Node {  
public:  
    int data;  
    Node* left;  
    Node* right;
```

```
    Node(int value) {  
        data = value;  
        left = right = nullptr;
```

```

    }
};

class BST {
private:
    Node* root;

public:
    BST() {
        root = nullptr;
    }

    void insert(int value) {
        root = insertNode(root, value);
    }

    Node* insertNode(Node* root, int value) {
        if (!root) return new Node(value);
        if (value < root->data) root->left = insertNode(root->left, value);
        else root->right = insertNode(root->right, value);
        return root;
    }

    int findMax(Node* root) {
        while (root->right) {
            root = root->right;
        }
        return root->data;
    }

    void displayMaxValue() {
        cout << findMax(root) << endl;
    }
};

int main() {
    BST tree;
    int N, value;

    // Read the number of nodes
    cin >> N;

```

```

// Insert elements into BST
for (int i = 0; i < N; i++) {
    cin >> value;
    tree.insert(value);
}

// Print the maximum value in BST
tree.displayMaxValue();

return 0;
}

int main() {
    int N, rootValue;
    scanf("%d", &N);

    struct TreeNode* root = NULL;

    for (int i = 0; i < N; i++) {
        int key;
        scanf("%d", &key);
        if (i == 0) rootValue = key;
        root = insert(root, key);
    }

    int maxVal = findMax(root);
    if (maxVal != -1) {
        printf("%d", maxVal);
    }

    return 0;
}

```

**Status :** Correct

**Marks :** 10/10