Rajalakshmi Engineering College

Name: Jhanani shree
Email: 240701215@rajalakshmi.edu.in
Roll no: 240701215
Phone: 7373333511
Branch: REC
Department: I CSE AH
Batch: 2028
Degree: B.E - CSE

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 : 0

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;
}
#include <stdio.h>
#include <stdlib.h>

// Define the structure for a tree node
struct TreeNode {
    int key;
    struct TreeNode* left;
    struct TreeNode* right;
};

// Function to insert a value into the BST
struct TreeNode* insert(struct TreeNode* root, int key) {
    // If the tree is empty, create a new node
```

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

    // Otherwise, recur down the tree
    if (key < root->key) {
        root->left = insert(root->left, key); // Insert in the left subtree
    } else {
        root->right = insert(root->right, key); // Insert in the right subtree
    }

    return root; // Return the unchanged root pointer
}

// Function to find the maximum value in the BST
int findMax(struct TreeNode* root) {
    if (root == NULL) {
        return -1; // Return -1 or some indication that the tree is empty
    }

    struct TreeNode* current = root;

    // Traverse to the rightmost node
    while (current->right != NULL) {
        current = current->right;
    }

    return current->key; // Return the maximum value
}

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 : Wrong **Marks : 0/10**