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

struct Node {
int data;
struct Node* left;
struct Node* right;
};


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



struct Node* insert(struct Node* root, int value) {
if (root == NULL) {
return createNode(value);
}

if (value < root->data) {
root->left = insert(root->left, value);
}
else if (value > root->data) {
root->right = insert(root->right, value);
}

return root;
}



int findHeight(struct Node* root) {
if (root == NULL) {
return -1;
}
int leftHeight = findHeight(root->left);
int rightHeight = findHeight(root->right);


return (leftHeight > rightHeight ? leftHeight : rightHeight) + 1;
}



int findDepth(struct Node* root, int key) {
int depth = 0;
```

```
while (root != NULL) {
if (key == root->data) {
return depth;
}
else if (key < root->data) {
root = root->left;
}
else {
root = root->right;
}
depth++;
}

return -1;
}



void inorder(struct Node* root) {
if (root != NULL) {
inorder(root->left);
printf("%d ", root->data);
inorder(root->right);
}
}


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


printf("Enter number of nodes to insert: ");
scanf("%d", &n);


printf("Enter value %d: ", i + 1);
scanf("%d", &value);
root = insert(root, value);
}


printf("\nInorder Traversal of BST: ");
inorder(root);
printf("\n");


int height = findHeight(root);
printf("Height of the tree = %d\n", height);


printf("Enter a node value to find its depth: ");
scanf("%d", &key);
```

```
int depth = findDepth(root, key);
if (depth != -1)
printf("Depth of node %d = %d\n", key, depth);
else
printf("Node %d not found in the tree.\n", key);

return 0;
}
```