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

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

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

struct Node* insert(struct Node* root, int key) {
    if (root == NULL)
        return newNode(key);
    if (key < root->data)
        root->left = insert(root->left, key);
    else
        root->right = insert(root->right, key);
    return root;
}

void kthMinUtil(struct Node* root, int k, int* count, int* result) {
    if (root == NULL || *count >= k)
        return;

    // Traverse left subtree
    kthMinUtil(root->left, k, count, result);

    (*count)++;
    if (*count == k) {
        *result = root->data;
        return;
    }

    // Traverse right subtree
    kthMinUtil(root->right, k, count, result);
}


int findKthMin(struct Node* root, int k) {
    int count = 0;
    int result = -1;
    kthMinUtil(root, k, &count, &result);
    return result;
```

```
}


int main() {
    struct Node* root = NULL;
    int n, val, k;

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

    printf("Enter node values:\n");
    for (int i = 0; i < n; i++) {
        scanf("%d", &val);
        root = insert(root, val);
    }

    printf("Enter value of k: ");
    scanf("%d", &k);

    int kthMin = findKthMin(root, k);
    if (kthMin == -1)
        printf("k is larger than the number of nodes in the BST.\n");
    else
        printf("The %d-th minimum value in the BST is: %d\n", k, kthMin);

    return 0;
}
```