Aim

To find the k-th minimum (smallest) value in a Binary Search Tree (BST).

Algorithm

1. Perform **inorder traversal** of the BST (gives sorted order).
2. Keep a counter while traversing.
3. When counter == k, return that node's value.

C Code

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

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

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

// Insert into BST
struct Node* insert(struct Node* root, int data) {
    if (root == NULL) return createNode(data);
    if (data < root->data)
        root->left = insert(root->left, data);
    else
```

```c
        root->right = insert(root->right, data);
    return root;
}

// Inorder traversal with counter
void kthMin(struct Node* root, int k, int *count, int *result) {
    if (root == NULL) return;

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

int main() {
    struct Node* root = NULL;
    root = insert(root, 20);
    root = insert(root, 8);
    root = insert(root, 22);
    root = insert(root, 4);
    root = insert(root, 12);
    root = insert(root, 10);
    root = insert(root, 14);

    int k = 3, count = 0, result = -1;
    kthMin(root, k, &count, &result);

    if (result != -1)
        printf("%d-th minimum value is: %d\n", k, result);
    else
        printf("Less than %d nodes in BST\n", k);

    return 0;
}
```

Input (Hardcoded BST)

BST = {20, 8, 22, 4, 12, 10, 14}
k = 3

Output

```
3-th minimum value is: 10
```

✅ **Result**
Program successfully finds the k-th minimum value in a BST.