```c
// C program to demonstrate insert
// operation in binary
// search tree.
#include <stdio.h>
#include <stdlib.h>

struct node {
    int key;
    struct node *left, *right;
};

// A utility function to create a new BST node
struct node* newNode(int item)
{
    struct node* temp
        = (struct node*)malloc(sizeof(struct node));
    temp->key = item;
    temp->left = temp->right = NULL;
    return temp;
}

// A utility function to do inorder traversal of BST
void inorder(struct node* root)
{
    if (root != NULL) {
        inorder(root->left);
        printf("%d \n", root->key);
        inorder(root->right);
    }
}

/* A utility function to insert
a new node with given key in
* BST */
struct node* insert(struct node* node, int key)
{
    /* If the tree is empty, return a new node */
    if (node == NULL)
        return newNode(key);
```

```
40:
41:     /* Otherwise, recur down the tree */
42:     if (key < node->key)
43:         node->left = insert(node->left, key);
44:     else if (key > node->key)
45:         node->right = insert(node->right, key);
46:
47:     /* return the (unchanged) node pointer */
48:     return node;
49: }
50:
51: // Driver Code
52: int main()
53: {
54:     /* Let us create following BST
55:               50
56:            /     \
57:            30    70
58:            / \ / \
59:     20 40 60 80 */
60:     struct node* root = NULL;
61:     root = insert(root, 50);
62:     insert(root, 30);
63:     insert(root, 20);
64:     insert(root, 40);
65:     insert(root, 70);
66:     insert(root, 60);
67:     insert(root, 80);
68:
69:     // print inoder traversal of the BST
70:     inorder(root);
71:
72:     return 0;
73: }
74:
```