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

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

struct node* createNode(int data){
    struct node *n; // creating a node pointer
    n = (struct node *) malloc(sizeof(struct node)); // Allocating
    n->data = data; // Setting the data
    n->left = NULL; // Setting the left and right children to NULL
    n->right = NULL; // Setting the left and right children to NUL
    return n; // Finally returning the created node
}

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

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

void inOrder(struct  node* root){
    if(root!=NULL){
        inOrder(root->left);
        printf("%d ", root->data);
        inOrder(root->right);
```

```
40:     }
41: }
42:
43: int isBST(struct  node* root){
44:     static struct node *prev = NULL;
45:     if(root!=NULL){
46:         if(!isBST(root->left)){
47:             return 0;
48:         }
49:         if(prev!=NULL && root->data <= prev->data){
50:             return 0;
51:         }
52:         prev = root;
53:         return isBST(root->right);
54:     }
55:     else{
56:         return 1;
57:     }
58: }
59:
60: struct node * search(struct node* root, int key){
61:     if(root==NULL){
62:         return NULL;
63:     }
64:     if(key==root->data){
65:         return root;
66:     }
67:     else if(key<root->data){
68:         return search(root->left, key);
69:     }
70:     else{
71:         return search(root->right, key);
72:     }
73: }
74:
75: int main(){
76:
77:     // Constructing the root node - Using Function (Recommended
78:     struct node *p = createNode(5);
```

```
79:     struct node *p1 = createNode(3);
80:     struct node *p2 = createNode(6);
81:     struct node *p3 = createNode(1);
82:     struct node *p4 = createNode(4);
83:     // Finally The tree looks like this:
84:     //       5
85:     //      / \
86:     //     3   6
87:     //    / \
88:     //   1   4
89:
90:     // Linking the root node with left and right children
91:     p->left = p1;
92:     p->right = p2;
93:     p1->left = p3;
94:     p1->right = p4;
95:
96:     struct node* n = search(p, 6);
97:     if(n!=NULL){
98:     printf("Found: %d", n->data);
99:     }
100:    else{
101:        printf("Element not found");
102:    }
103:    return 0;
104: }
105:
```