

```

1: #include<stdio.h>
2: #include<malloc.h>
3:
4: struct node{
5:     int data;
6:     struct node* left;
7:     struct node* right;
8: };
9:
10: struct node* createNode(int data){
11:     struct node *n; // creating a node pointer
12:     n = (struct node *) malloc(sizeof(struct node)); // Allocating
13:     n->data = data; // Setting the data
14:     n->left = NULL; // Setting the left and right children to NULL
15:     n->right = NULL; // Setting the left and right children to NULL
16:     return n; // Finally returning the created node
17: }
18:
19: void preOrder(struct node* root){
20:     if(root!=NULL){
21:         printf("%d ", root->data);
22:         preOrder(root->left);
23:         preOrder(root->right);
24:     }
25: }
26:
27: void postOrder(struct node* root){
28:     if(root!=NULL){
29:         postOrder(root->left);
30:         postOrder(root->right);
31:         printf("%d ", root->data);
32:     }
33: }
34:
35: void inOrder(struct node* root){
36:     if(root!=NULL){
37:         inOrder(root->left);
38:         printf("%d ", root->data);
39:         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 * searchIter(struct node* root, int key){
61:     while(root!=NULL){
62:         if(key == root->data){
63:             return root;
64:         }
65:         else if(key<root->data){
66:             root = root->left;
67:         }
68:         else{
69:             root = root->right;
70:         }
71:     }
72:     return NULL;
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 = searchIter(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:

```