```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);
    }
}

int isBST(struct  node* root){
    static struct node *prev = NULL;
    if(root!=NULL){
        if(!isBST(root->left)){
            return 0;
        }
        if(prev!=NULL && root->data <= prev->data){
            return 0;
        }
        prev = root;
        return isBST(root->right);
    }
    else{
        return 1;
    }
}

struct node * searchIter(struct node* root, int key){
    while(root!=NULL){
        if(key == root->data){
            return root;
        }
        else if(key<root->data){
            root = root->left;
        }
        else{
            root = root->right;
        }
    }
    return NULL;
}

void insert(struct node *root, int key){
   struct node *prev = NULL;
   while(root!=NULL){
       prev = root;
```

```
        if(key==root->data){
            printf("Cannot insert %d, already in BST", key);
            return;
        }
        else if(key<root->data){
            root = root->left;
        }
        else{
            root = root->right;
        }
    }
    struct node* new = createNode(key);
    if(key<prev->data){
        prev->left = new;
    }
    else{
        prev->right = new;
    }

}

int main(){

    // Constructing the root node - Using Function (Recommended
    struct node *p = createNode(5);
    struct node *p1 = createNode(3);
    struct node *p2 = createNode(6);
    struct node *p3 = createNode(1);
    struct node *p4 = createNode(4);
    // Finally The tree looks like this:
    //      5
    //     / \
    //    3   6
    //   / \
    //  1   4

    // Linking the root node with left and right children
    p->left = p1;
    p->right = p2;
```

```c
    p1->left = p3;
    p1->right = p4;

    insert(p, 16);
    printf("%d\n", p->right->right->data);
    inOrder(p);
    return 0;
}
```