```
//By Ritwik Chandra Pandey
//On 2 Sep 2021
//BST Inorder!Recursion

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

struct node {
	int data;
	struct node *left, *right;
};
typedef struct node *BSTNODE;

struct stacknode {
	BSTNODE node;
	struct stacknode * next;
};

typedef struct stacknode * STKNODE;
STKNODE top = NULL;
int isempty() {
	if(top == NULL) {
		return 1;
	}
	return 0;
}
void push(BSTNODE b) {
	STKNODE temp;
	temp = (STKNODE)malloc(sizeof(struct stacknode));
	if(temp == NULL) {
		printf("Stack is overflow.\n");
	} else {
		temp -> node = b;
		temp -> next = top;
		top = temp;
	}
}
BSTNODE peek() {
	if (top == NULL) {
		return NULL;
```

```
        }
        return top->node;
}
BSTNODE pop() {
        STKNODE temp;
        BSTNODE b;
        if(top == NULL) {
                printf("Stack is underflow.\n");
        } else {
                temp = top;
                top = top -> next;
                b = temp->node;
                free(temp);
                return b;
        }
}

STKNODE newStackNode(BSTNODE b) {
        STKNODE temp =  (STKNODE)malloc(sizeof(struct node));
        temp->node = b;
        temp->next = NULL;
        return temp;
}
BSTNODE newNodeInBST(int item) {
        BSTNODE temp =  (BSTNODE)malloc(sizeof(struct node));
        temp->data = item;
        temp->left = temp->right = NULL;
        return temp;
}
void inorderInBST(BSTNODE root) {
        BSTNODE curr;
        curr = root;
        while(1){
                if(curr!=NULL){
                        push(curr);
                        curr=curr->left;
                }else{
                        curr=pop();
                        printf("%d ",curr->data);
                        curr = curr->right;
```

```
                }
                if(top==NULL && curr==NULL) return;
        }
}
BSTNODE insertNodeInBST(BSTNODE node, int ele) {
        if (node == NULL) {
                printf("Successfully inserted.\n");
                return newNodeInBST(ele);
        }
        if (ele < node->data)
                node->left  = insertNodeInBST(node->left,ele);
        else if (ele > node->data)
                node->right = insertNodeInBST(node->right,ele);
        else
                printf("Element already exists in BST.\n");
        return node;
}
BSTNODE minValueNode(BSTNODE node) {
        BSTNODE temp = node;
        while(temp->left!=NULL){
                temp = temp->left;
        }
        return temp;

}
BSTNODE deleteNodeInBST(BSTNODE root, int ele) {
        if (root == NULL) {
                printf("Cannot find %d in the binary search tree.\n",ele);
                return root;
        }
        if (ele < root->data)
                root->left = deleteNodeInBST(root->left,ele);
        else if (ele > root->data)
                root->right = deleteNodeInBST(root->right,ele);
        else {
                if (root->left == NULL) {
                        BSTNODE temp = root->right;
                        printf("Deleted %d from binary search tree.\n",ele);
                        free(root);
                        return temp;
```

```
                }
                else if (root->right == NULL) {
                        BSTNODE temp = root->left;
                        printf("Deleted %d from binary search tree.\n",ele);
                        free(root);
                        return temp;
                }
                BSTNODE temp = minValueNode(root->right);
                root->data = temp->data;
                temp->data = ele;
                root->right = deleteNodeInBST(root->right,ele);
    }
        return root;
}
void main() {
        int x, op;
        BSTNODE root = NULL;
        while(1) {
                printf("1.Insert 2.Delete 3.Inorder Traversal 4.Exit\n");
                printf("Enter your option : ");
                scanf("%d", &op);
                switch(op) {
                        case 1:printf("Enter an element to be inserted : ");
                                scanf("%d", &x);
                                root = insertNodeInBST(root,x);
                                break;
                        case 2:printf("Enter an element to be deleted : ");
                                scanf("%d", &x);
                                root = deleteNodeInBST(root,x);
                                break;
                        case 3:
                                if(root == NULL) {
                                        printf("Binary Search Tree is empty.\n");
                                }
                                else {
                                        printf("Elements of the BST (in-order traversal): ");
                                        inorderInBST(root);
                                        printf("\n");
                                }
                                break;
```

```
                case 4:
                        exit(0);
            }
        }
}
```