```
//Created By Ritwik Chandra Pandey
//On 25 Oct 2021
//Implementation of AVL tree - search and postorder traversal


#include<stdio.h>
#include<conio.h>

struct node {
	int data;
	struct node *left,*right;
	int ht;
};
typedef struct node * AVLNODE;
AVLNODE createNodeInAVL(int item) {
	AVLNODE temp =  (AVLNODE)malloc(sizeof(struct node));
	temp->data = item;
	temp->left = temp->right = NULL;
	return temp;
}
int height(AVLNODE root) {
	int lh,rh;
	if(root==NULL)
		return(0);
	if(root->left==NULL)
		lh=0;
	else
		lh=1+root->left->ht;
	if(root->right==NULL)
		rh=0;
	else
		rh=1+root->right->ht;
	if(lh>rh)
		return(lh);
	return(rh);
}

AVLNODE rotateRight(AVLNODE x) {
	AVLNODE y;
```

```
        y=x->left;
        x->left=y->right;
        y->right=x;
        x->ht=height(x);
        y->ht=height(y);
        return(y);
}
AVLNODE rotateLeft(AVLNODE x) {
        AVLNODE y;
        y=x->right;
        x->right=y->left;
        y->left=x;
        x->ht=height(x);
        y->ht=height(y);
        return(y);
}
AVLNODE LL(AVLNODE root) {
        root=rotateLeft(root);
        return(root);
}
AVLNODE RR(AVLNODE root) {
        root=rotateRight(root);
        return(root);
}
AVLNODE LR(AVLNODE root) {
        root->left=rotateLeft(root->left);
        root=rotateRight(root);
        return(root);
}
AVLNODE RL(AVLNODE root) {
        root->right=rotateRight(root->right);
        root=rotateLeft(root);
        return(root);
}
int balancefactor(AVLNODE root) {
        int lh,rh;
        if(root==NULL)
                return(0);
        if(root->left==NULL)
                lh=0;
```

```
	else
		lh=1+root->left->ht;
	if(root->right==NULL)
		rh=0;
	else
		rh=1+root->right->ht;
	return(lh-rh);
}
void postorderInAVL(AVLNODE root) {
	if(root!=NULL){
		postorderInAVL(root->left);
		postorderInAVL(root->right);
		printf("%d(%d) ",root->data, balancefactor(root));
	}
}

AVLNODE insertNodeInAVL(AVLNODE root,int x) {
	if(root==NULL) {
		root=createNodeInAVL(x);
		printf("Successfully inserted.\n");
	}
	else if(x > root->data) {
		root->right=insertNodeInAVL(root->right,x);
		if(balancefactor(root)==-2)
			if(x>root->right->data)
				root=LL(root);
			else
				root=RL(root);
	}
	else if(x<root->data) {
		root->left=insertNodeInAVL(root->left,x);
		if(balancefactor(root)==2)
			if(x < root->left->data)
				root=RR(root);
			else
				root=LR(root);
	}
	else {
		printf("Element %d already exists.\n",x);
	}
```

```
	root->ht=height(root);
	return(root);
}

AVLNODE searchNodeInAVL(AVLNODE root, int ele) {
	if(root==NULL || root->data==ele){
		return root;
	}
	if(root->data<ele){
		return searchNodeInAVL(root->right,ele);
	}
	else{return searchNodeInAVL(root->left,ele);
}}


int main() {
	int x, op;
	AVLNODE root = NULL;
	while(1) {
		printf("1.Insert 2.Search 3.Postorder 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 = insertNodeInAVL(root,x);
				break;
			case 2:
				printf("Enter an element to be searched : ");
				scanf("%d", &x);
				if( searchNodeInAVL(root,x) == NULL)
					printf("Element not found in the AVL tree.\n");
				else
					printf("Element found in the AVL tree.\n");
				break;
			case 3:
				if(root == NULL) {
					printf("AVL Tree is empty.\n");
				}
				else {
```

```
                                        printf("Elements of the AVL tree (post-order traversal): ");
                                        postorderInAVL(root);
                                        printf("\n");
                                }
                                break;
                        case 4:exit(0);
                }
        }
}
```