```
//Created by Ritwik Chandra Pandey on 27 Sep 2021
//Implementation of AVL - Rotations, Insertion, Inorder

#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;
	else
	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);
}

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 RR(AVLNODE root) {
        root = rotateRight(root);
        return root;
}

AVLNODE LL(AVLNODE root) {
        root = rotateLeft(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 inorderInAVL(AVLNODE root) {
	if(root!=NULL){
		inorderInAVL(root->left);
		printf("%d(%d) ", (root->data), balancefactor(root));
		inorderInAVL(root->right);
	}
}

AVLNODE insertNodeInAVL(AVLNODE root,int x) {
	if (root==NULL){
		root = createNodeInAVL(x);
		printf("Successfully inserted.\n");
	}
	else if(root->data < x){
		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 already exists.");
}
root->ht = height(root);
return root;
}

int main() {
		int x, op;
		AVLNODE root = NULL;
		while(1) {
			printf("1.Insert 2.Inorder Traversal 3.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:
					if(root == NULL) {
						printf("AVL Tree is empty.\n");
					}
					else {
						printf("Elements of the AVL tree (in-order traversal): ");
						inorderInAVL(root);
						printf("\n");
					}
					break;
```

```
                    case 3:exit(0);
                }
        }
}
```