```
//avlIf.h
struct avl{
	int data;
	struct avl *left;
	struct avl *right;
	int ht;
};

static int height(struct avl *t);
struct avl* tmax(struct avl *t);
struct avl* tmin(struct avl *t);
struct avl* insert(struct avl* t,int x);
struct avl* SRleft(struct avl* k2);
struct avl* SRright(struct avl* k2);
struct avl* DRleft(struct avl* k3);
struct avl* DRright(struct avl* k3);
struct avl* find(struct avl *t, int x);
void display(struct avl* t,int depth);

//avlImpl.h
static int height(struct avl *t){
	if(t==NULL)
		return -1;
	else
		return t->ht;
}

int max(int a,int b){
	if(a>b)
		return a;
	else return b;
```

```
}

struct avl* tmin(struct avl *t)
{
	if(t==NULL)
		return NULL;
	else if(t->left==NULL)
	{
		return t;
	}
	else
		return tmin(t->left);
}

struct avl* tmax(struct avl *t)
{
	if(t==NULL)
	{
		return NULL;
	}
	else if(t->right==NULL)
	{
		return t;
	}
	else
	{
		return tmax(t->right);
	}
}

struct avl* insert(struct avl* t,int x){
```

```
//printf("Insert called");
int htdiff;
struct avl* temp;
if(t==NULL){
        t=(struct avl*)malloc(sizeof(struct avl));
        //printf("\nCREATING NULL");
        t->data=x;
        t->left=t->right=NULL;
        t->ht=0;
}

else if(x<t->data){
        t->left=insert(t->left,x);
        htdiff=abs(height(t->left)-height(t->right));
        if(htdiff==2){
                if(x<(t->left)->data)
                        t=SRleft(t);
                else
                        t=DRleft(t);
        }
}

else if(x>t->data){
        t->right=insert(t->right,x);
        htdiff=abs(height(t->left)-height(t->right));
        //printf("\nhtdiff=%d",htdiff);
        if(htdiff==2){
                if(x>(t->right)->data)
                        t=SRright(t);
                else
                        t=DRright(t);
```

```
                }
        }

        t->ht=max(height(t->left),height(t->right))+1;
        //printf("\nht of %d is %d",t->data,t->ht);
        //printf("\nReturning t");
        return t;
}

struct avl* SRleft(struct avl* k2){
        struct avl* k1=k2->left;
        k2->left=k1->right;
        k1->right=k2;

        k2->ht=max(height(k2->left),height(k2->right))+1;
        k1->ht=max(height(k1->left),height(k2->right))+1;
        return k1;
}

struct avl* SRright(struct avl* k2){
        struct avl* k1=k2->right;
        k2->right=k1->left;
        k1->left=k2;

        k2->ht=max(height(k2->left),height(k2->right))+1;
        k1->ht=max(height(k1->left),height(k2->right))+1;
        return k1;
}

struct avl* DRleft(struct avl* k3){
        k3->left=SRright(k3->left);
```

```
        return SRleft(k3);
}

struct avl* DRright(struct avl* k3){
        k3->right=SRright(k3->left);
        return SRleft(k3);
}

struct avl* find(struct avl *t, int x)
{
        if(t==NULL)
        {
                return NULL;
        }
        if(x<t->data)
        {
                return find(t->left,x);
        }
        if(x>t->data)
        {
                return find(t->right,x);
        }
        else
                return t;
}

void display(struct avl *t,int depth){
        int i;
        for(i=0;i<depth;i++)
                printf("\t");
        printf("%d",t->data);
```

```
        printf("\n");
        if(t->left!=NULL)
        display(t->left,depth+1);
        if(t->right!=NULL)
        display(t->right,depth+1);
}

//avlAppl.c
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<string.h>
#include "avlIf.h"
#include "avlImpl.h"

void main(){
        struct avl *t=NULL;
        int i;
   int n;

        for(i=0;i<6;i++){
                printf("Number %d: ",i+1);
                scanf("%d",&n);
                t=insert(t,n);
                printf("\n");
                display(t,0);
        }
   printf("Enter element to search: ");
   scanf("%d",&n);
   struct avl* found=find(t,n);
   if(found==NULL)
```

```
        printf("%d found",n);
    else
        printf("Element not found");
}

/*
OUTPUT:
Number 1: 34

34
Number 2: 14

34
        14
Number 3: 4

14
        4
        34
Number 4: 24

14
        4
        34
                24
Number 5: 32

14
        4
        32
                24
```

34

Number 6: 37

32

14

4

24

34

37

Enter element to search: 4

4 found

*/