```
//S Harshini-185001058

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

struct node
{
	char word[25];
	char tamil[25],hindi[25];
	struct node *left,*right;
	int height;
};

typedef struct node *position;
typedef struct node *avltree;

static int height(position p)
{
	if(p==NULL)
		return -1;
	else
		return p->height;
}

int max(int x,int y)
{
	if(x>y)
		return x;
	return y;
}

static position singlerotatewithleft(position k2)
{
	printf("Single Rotate with Left\n");
	position k1;
	k1=k2->left;
	k2->left=k1->right;
	k1->right=k2;
	k2->height=max(height(k2->left),height(k2->right))+1;
	k1->height=max(height(k1->left),height(k1->right))+1;
	return k1;
}
```

```
static position singlerotatewithright(position k2)
{
	printf("Single Rotate with Right\n");
	position k1;
	k1=k2->right;
	k2->right=k1->left;
	k1->left=k2;
	k2->height=max(height(k2->left),height(k2->right))+1;
	k1->height=max(height(k1->left),height(k1->right))+1;
	return k1;
}

static position doublerotatewithleft(position k3)
{
	printf("Double Rotate with Left contains:\n");
	k3->left=singlerotatewithright(k3->left);
	return singlerotatewithleft(k3);
}

static position doublerotatewithright(position k3)
{
	printf("Double Rotate with Right contains:\n");
	k3->right=singlerotatewithleft(k3->right);
	return singlerotatewithright(k3);
}

avltree insert(char *word,char *tamil,char *hindi,avltree a)
{
	if(a==NULL)
	{
		a=(avltree)malloc(sizeof(struct node));
		if(a==NULL)
			printf("Out of Space");
		else
		{
			a->height=0;
			strcpy(a->word,word);
			strcpy(a->tamil,tamil);
			strcpy(a->hindi,hindi);
			a->left=a->right=NULL;
		}
	}
```

```
        else if(strcmp(word,a->word)<0)
        {
                a->left=insert(word,tamil,hindi,a->left);
                if(height(a->left)-height(a->right) ==2)
                        if( strcmp(word,a->left->word)<0 )
                                a=singlerotatewithleft(a);
                        else
                                a=doublerotatewithleft(a);
        }
        else if( strcmp(word,a->word)>0 )
        {
                a->right=insert(word,tamil,hindi,a->right);
                if(height(a->right)-height(a->left) ==2)
                        if( strcmp(word,a->right->word)>0 )
                                a=singlerotatewithright(a);
                        else
                                a=doublerotatewithright(a);
        }
        a->height=max( height(a->left),height(a->right)) +1;
        return a;
}

void inorder(avltree a)
{
        if(a!=NULL)
        {
                inorder(a->left);
                printf("%s ",a->word);
                inorder(a->right);
        }
}


void printtree(avltree a,int space)
{
        if(a==NULL)
                return;
        space+=10;
        printtree(a->right,space);
        printf("\n");
        for(int i=10;i<space;i++)
                printf(" ");
        printf("%s\n",a->word);
        printtree(a->left,space);
}

void printLang(char *word,avltree a,int *found)
{
        if(a!=NULL)
        {
                printLang(word,a->left,&*found);
                if( strcmp(a->word,word)==0)
                {
                        printf("Tamil trans:%s\nHindi trans:%s",a->tamil,a->hindi);
                        *found=1;
                }
                printLang(word,a->right,&*found);
        }
}

int main()
{
        avltree a=NULL;
        char word[20],tamil[25],hindi[25];
        printf("Enter Elements to the tree Enter -1 to stop\n");

        do{
                printf("enter word:");
                scanf(" %[^\n]",word);
                if( (strcmp(word,"-1"))==0 )
                        break;
                printf("enter in tamil:");
                scanf(" %[^\n]",tamil);
                printf("enter in hindi:");
                scanf(" %[^\n]",hindi);
                a=insert(word,tamil,hindi,a);
                printtree(a,0);
        }while(1);

        printf("\ndictionary enter -1 to exit\n");
        do
        {
                printf("\nEnter English Word:");
                scanf(" %[^\n]",word);
                if( (strcmp(word,"-1"))==0 )
                        break;
                int found=0;
                printLang(word,a,&found);
                if(found==0)
                        printf("Not Available in Dictionary");
        }while(1);

        return 0;
}
```