```c
/*
 * Joao Paulo Batista Ferreira
 * 2009113274
 * Algoritmos e Estruturas de Dados - TP3 exC
 */

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

typedef struct item{
    unsigned int counter;
    char word[256];
}item;

typedef struct node{
    item info;
    char color;
    struct node* right;
    struct node* left;
}node;

typedef node* nodePtr;

int isRed(nodePtr leaf)
{
    if ( (leaf == NULL) || (leaf->color=='B') )
        return 0;
    else
        return 1;

}

void eraseTree(nodePtr leaf)
{
    if (leaf == NULL)
        return;
    eraseTree(leaf->left);
    eraseTree(leaf->right);
    free(leaf);
}

void visit (nodePtr leaf)
{
    printf("%s %d\n", leaf->info.word, leaf->info.counter);
}

void emOrdem (nodePtr leaf)
{
    if (leaf != NULL)
    {
        emOrdem(leaf->left);
        visit (leaf);
        emOrdem(leaf->right);
    }
}
```

```
nodePtr leftRotation (nodePtr leaf)
{
    nodePtr aux;

    aux = leaf->right;
    leaf->right = aux->left;
    aux->left = leaf;

    if (aux != NULL)
        aux->color = 'B';
    if (aux->left != NULL)
        aux->left->color = 'R';
    if (aux->right != NULL)
        aux->right->color = 'R';

    return aux;
}

nodePtr rightRotation (nodePtr leaf)
{
    nodePtr aux;

    aux = leaf->left;
    leaf->left = aux->right;
    aux->right = leaf;

    if (aux != NULL)
        aux->color = 'B';
    if (aux->left != NULL)
        aux->left->color = 'R';
    if (aux->right != NULL)
        aux->right->color = 'R';

    return aux;
}

nodePtr insert (nodePtr leaf, char* word, int location)
{

    if (leaf == NULL)
    {
        leaf = (nodePtr) malloc (sizeof(node));

        strcpy(leaf->info.word, word);
        leaf->info.counter = 1;
        leaf->color = 'R';
        leaf->left = NULL;
        leaf->right = NULL;

        return leaf;
    }

    if( isRed(leaf->left) && isRed(leaf->right) )
    {
        leaf->color = 'R';
        leaf->left->color = leaf->right->color = 'B';
```

```
    }

    if( strcmp(word, leaf->info.word) > 0)
    {
        leaf->right = insert(leaf->right, word, 1);

        if( isRed(leaf) && isRed(leaf->right) && !location )
            leaf = leftRotation(leaf);

        if( isRed(leaf->right) && isRed(leaf->right->right) )
        {
            leaf = leftRotation(leaf);
            leaf->color = 'B';
            leaf->left->color = 'R';
        }
    }

    else if( strcmp(word, leaf->info.word) < 0)
    {
        leaf->left = insert(leaf->left, word,0);

        if( isRed(leaf) && isRed(leaf->left) && location)
            leaf = rightRotation(leaf);

        if( isRed(leaf->left) && isRed(leaf->left->left) )
        {
            leaf = rightRotation(leaf);
            leaf->color = 'B';
            leaf->right->color = 'R';
        }

    }

    else
        leaf->info.counter++;

    return leaf;
}

nodePtr worker(char *word, nodePtr leaf)
{
    int i, size;

    size = strlen(word);

    for (i=0 ; i<size ; i++)
        word[i]=tolower(word[i]);

    leaf = insert(leaf, word, 0);

    leaf->color = 'B';

    return leaf;
}

int main(int argc, char **argv)
{
```

```
    char word[256];
    nodePtr tree=NULL;

    while( (scanf("%s", word)) != EOF )
        tree = worker(word, tree);

    emOrdem(tree);

    eraseTree(tree);

    return 0;
}
```