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

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

#define N_MAX_LINKS 13

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

struct node {
       item info;
       struct node** next;
       int size;
}node;

typedef struct node* nodePtr;

nodePtr list;
int n_items, n_lvl;

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

void printList(nodePtr list)
{
    if (list != NULL)
    {
        visit(list);
        printList(list->next[0]);
    }
}

void freeMem()
{
    nodePtr next=NULL;
    nodePtr x = list->next[0];
    while(x != NULL)
    {
        next=x->next[0];
        free(x);
        x = next;
    }
}

int less(nodePtr sl, char* w, int k)
{
    if(sl->next[k] == NULL )
    {
        return 1;
    }

    else if( strcmp(w, sl->next[k]->info.word) < 0 )
    {
        return 1;
    }

    else
    {
        return 0;
    }
}

int searchList(nodePtr sl, char* word, int k)
{
    if(sl == NULL)
        return 0;

    else if( strcmp(word, sl->info.word) == 0)
    {
        sl->info.counter++;
        return 1;
    }

    else if(less(sl, word, k))
    {
        if(k == 0)
        {
            return 0;
        }
        return searchList(sl, word, k-1);
    }

    return searchList(sl->next[k], word, k);
}

int searchNumber(char* word)
{
    return searchList(list, word, n_lvl);
}

void addNode(nodePtr sl, nodePtr x, int k)
{
    if(less(sl,x->info.word,k))
    {
        if(sl != NULL && x != NULL)
        {
                if(k < x->size)
                {
                    x->next[k] = sl->next[k];
                    sl->next[k] = x;
                }
        }
```

```c
        if(k == 0)
        {
            return;
        }
        addNode(sl,x,k-1);
        return;
    }
    addNode(sl->next[k],x,k);
}

int randX()
{
    int i,j,t;
    t = rand();
    for(i = 1, j = 2; i<N_MAX_LINKS; i++, j += j)
    {
          if(t > RAND_MAX/j)
          {
               break;
          }
    }

    if(i > n_lvl)
    {
         n_lvl = i;
    }

    return i;
}

nodePtr createNode(char* word, int k)
{
    int i;
    nodePtr x = (nodePtr) malloc (sizeof(node));
    x->next = (nodePtr*) malloc (k*sizeof(node));
    strcpy(x->info.word, word);
    x->info.counter = 1;
    x->size = k;
    for(i = 0; i<k; i++)
        x->next[i] = NULL;

    return x;
}

void initList(int n_max_nos)
{
    n_items = n_lvl = 0;
    list = createNode("-1", n_max_nos);
}

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

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

    if ( searchNumber(word) == 0 )
    {
        aux = createNode(word, randX());
        addNode(list, aux, n_lvl);
    }
}

int main()
{
    initList(N_MAX_LINKS);
    char word[256];

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

    printList(list->next[0]);

    freeMem();

    return 0;
}
```