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

#define MAX_M 10000
//#define P 256

typedef struct BSTNode{
    char* key;// key, word
    int count;// number of occurrences
    struct BSTNode* left;
    struct BSTNode* right;
}TNode;

TNode* bst[MAX_M];
int rs;
int addmod(int a, int b, int P){
    a = a % P;
    b = b % P;
    int tmp = P - a;
    if(tmp > b) return a+b;
    return b - tmp;
}
int mulmod(int a, int b, int P){
    if(a == 0 || b == 0) return 0;
    if(a == 1) return b%P;
    if(b == 1) return a%P;
    if(a > b){
        int tmp = a; a = b; b = tmp;
    }
    int c = mulmod(a/2,b,P);
    c = addmod(c,c,P);
    if(a%2==0) return c;
    else return addmod(c,b,P);
}
int XmuN(int X, int N,int P){
    if(N == 1) return X%P;
    int a = XmuN(X,N/2,P);
    a = mulmod(a,a,P);
    if(N%2 == 0) return a;
    else return mulmod(a,X,P);
}

TNode* makeNode(char* key){
    TNode* p = (TNode*)malloc(sizeof(TNode));
    p->left = NULL;
    p->right = NULL;
    p->count = 1;
    p->key = key;
}
int hash(char* key){
    // TODO
    int n = strlen(key);
    //printf("n = %d\n",n);
    int i;
    int h = 0;
    for(i = 0; i < n; i++){
        h = addmod(mulmod(h,256,MAX_M),key[i],MAX_M);
    }
    return h;
}
void init(){
    int i;
    for(i = 0; i < MAX_M; i++)
        bst[i] = NULL;
}
int size(TNode* r){
    if(r == NULL) return 0;
    return 1 + size(r->left) + size(r->right);
}
void print(TNode* r){
    if(r == NULL) return;
    print(r->left);
    printf("%s: %d\n",r->key,r->count);
    print(r->right);
}
TNode* searchBST(TNode* r, char* key){
    // TODO
    if(r == NULL) return NULL;
    if(strcmp(r->key,key) == 0){
        return r;
    }
    if(strcmp(r->key,key) > 0){
        return searchBST(r->left,key);
    }
    return searchBST(r->right,key);
}
TNode* addNode(TNode* r, char* key){
    if(r == NULL) return makeNode(key);
    if(strcmp(r->key,key) > 0)
        r->left = addNode(r->left,key);
    else
        r->right = addNode(r->right,key);
    return r;
}

TNode* addBST(TNode* r, char* key){
    TNode* p = searchBST(r,key);
    if(p != NULL){
        //printf("addBST(%s), p != NULL\n",key);
        p->count = p->count + 1;
    }else{
        //printf("addBST(%s), p == NULL\n",key);
        r = addNode(r,key);
    }
    return r;
}
TNode* search(char* key){
    int h = hash(key);
    return searchBST(bst[h],key);
}
void addDict(char* key){
    // TODO
    int h = hash(key);
    bst[h] = addBST(bst[h],key);
}
void solve(){
    rs = 0;
    while(1){
        char* s = (char*)malloc(1000* sizeof(char));
        //if(scanf("%s",s) == EOF) break;
        scanf("%s",s);
        if(strcmp(s,"-1") == 0) break;
        //printf("%s, %d",s,strlen(s));
        int c = 0;
        TNode* nod = search(s);
        if(nod != NULL) c = nod->count;
        if(c + 1 > rs) rs = c+1;
        addDict(s);
        //printf("added %s, h = %d\n",s,hash(s));
    }
    printf("%d",rs);

}
void solveFromFile(char* filename, char* fo){
    FILE* f = fopen(filename,"r");
    //char s[10000];
    rs = 0;
    while(1){
        char* s = (char*)malloc(1000* sizeof(char));
        //if(fscanf(f,"%s",s) == EOF) break;
        fscanf(f,"%s",s);
        if(strcmp(s,"-1") == 0) break;
        //printf("%s, %d\n",s,strlen(s));
        int c = 0;
        TNode* nod = search(s);
        if(nod != NULL) c = nod->count;
        if(c + 1 > rs) rs = c+1;
        addDict(s);
        //printf("added %s, h = %d\n",s,hash(s));
    }
    fclose(f);

    f = fopen(fo,"w");
    fprintf(f,"%d",rs);
    fclose(f);
    //printf("%d\n",rs);
}
void printDict(){
    int i;
    for(i = 0; i < MAX_M; i++){
        if(size(bst[i]) > 0){
            print(bst[i]);
        }
        //printf("bst[%d].sz = %d\n",i,size(bst[i]));
    }
}
void createTest(char* fi, char* fo, int N, int minLen, int maxLen){
    char* T = "abcdefghijklmnopqrstuvwxyz0123456789";
    FILE* f = fopen(fi,"w");
    int i;
    srand(time(NULL));
    for(i = 1; i <= N; i++){
        int L = rand()%(maxLen-minLen+1) + minLen;
        int j;
        for(j = 1; j <= L ;j++){
            int idx = rand()%strlen(T);
            fprintf(f,"%c",T[idx]);
        }
        fprintf(f,"    ");
    }
    fprintf(f,"\n  -1");
    fclose(f);
    solveFromFile(fi,fo);
}

int main(){
    //printf("START\n");
    solveFromFile("Test09/DICTIONARY.INP",
                  "Test09/DICTIONARY.OUT");
    //createTest("Test00/DICTIONARY.INP",
    //           "Test00/DICTIONARY.OUT",30,1,1);

    //solve();
    //printDict();
    //printf("%d\n",hash("abc"));
}
```