最常见的单词

给定一个段落 (paragraph) 和一个禁用单词列表 (banned)。返回出现次数最多，同时不在禁用列表中的单词。

题目保证至少有一个词不在禁用列表中，而且答案唯一。

禁用列表中的单词用小写字母表示，不含标点符号。段落中的单词不区分大小写。答案都是小写字母。

示例：

输入：

```
paragraph = "Bob hit a ball, the hit BALL flew far after it was hit."

banned = ["hit"]
```

输出： "ball"

解释：

"hit" 出现了 3 次，但它是一个禁用的单词。

"ball" 出现了 2 次 (同时没有其他单词出现 2 次)，所以它是段落里出现次数最多的，且不在禁用列表中的单词。

注意，所有这些单词在段落里不区分大小写，标点符号需要忽略（即使是紧挨着单词也忽略，比如 "ball,"），

"hit"不是最终的答案，虽然它出现次数更多，但它在禁用单词列表中。

解题思路
对字典预处理，大写转小写，其他字符转空格；
将字典的单词保存到结构体数组中，并统计出现次数；
按照出现次数对结构体数组降序排序；
逐一对比结构体数组，第一个不在禁用列表里的即出现次数最多的。

```
#define MAX_WORD_NUM 1000

typedef struct {
    char *word;
    int count;
} Word;
```

```
Word result[MAX_WORD_NUM];       // 保存字典单词的结构体数组
int g_diff_word_cnt;   // 保存在结构体数组中不同单词的数量

/* 对字典预处理，大写转小写，其他字符转空格 */
void initPara(char *paragraph)
{
    int len = strlen(paragraph);
    int i;
    for (i = 0; i < len; i++) {
        if (paragraph[i] == '!' || paragraph[i] == '?' ||
            paragraph[i] == '\'' || paragraph[i] == ',' ||
            paragraph[i] == ';' || paragraph[i] == '.') {
            paragraph[i] = ' ';
        } else if (paragraph[i] >= 'A' && paragraph[i] <= 'Z') {
            paragraph[i] = tolower(paragraph[i]);
        }
    }
}

/* 将字典的单词保存到结构体数组中 */
void SaveWord(char *str)
{
    int i;
    for (i = 0; i < g_diff_word_cnt; i++) {
        if (strcmp(result[i].word, str) == 0) {  // 如果当前数组与结构体数组
中的相同，则计数加 1
            result[i].count++;
            return;
        }
    }

    result[g_diff_word_cnt].word = str;  // 如果当前数组在结构体数组中的没有，
则加入结构体数组
    result[g_diff_word_cnt].count = 1;

    g_diff_word_cnt++;  // 结构体数组中不同单词的数量加 1
}

/* 判断当前单词是否在禁用列表里 */
bool isInBanned(char *str, char **banned, int bannedSize)
{
    int i;
    for (i = 0; i < bannedSize; i++) {
        if (strcmp(str, banned[i]) == 0) {
```

```
            return true;
        }
    }

    return false;
}

int compare(const void *a, const void *b)
{
    Word *pa = (Word*)a;
    Word *pb = (Word*)b;

    return pb->count - pa->count;  // 将结构体数组中的不同单词按照出现次数降序排列
}

char *mostCommonWord(char *paragraph, char **banned, int bannedSize)
{
    initPara(paragraph);
    memset(result, 0, sizeof(Word) * MAX_WORD_NUM);
    g_diff_word_cnt = 0;

    char *p = strtok(paragraph, " ");  // 使用空格分隔字典，将分割后的单词保存到结构体数组
    while (p != NULL) {
        SaveWord(p);
        p = strtok(NULL, " ");
    }

    qsort(result, g_diff_word_cnt, sizeof(Word), compare);

    int i;
    for (i = 0; i < g_diff_word_cnt; i++) {
        if (!isInBanned(result[i].word, banned, bannedSize)) {  // 结构体数组中第一个不在禁用列表里的单词即出现次数最多的
            return result[i].word;
        }
    }

    return NULL;
}
```