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

typedef struct {
    char* name;
    int count;
} kwarg;kwarg* init_table(char** arg_names, int num_args);char**
parse(char* line, int len);void count(char** tokens, kwarg* table, int
row_len, int col_len);int word_counter(char* str, int chars);
//https://en.cppreference.com/w/c/language/compound_literal
//how to make a literal struct
kwarg* init_table(char** arg_names, int num_args) {
    kwarg* head = (kwarg*) malloc(num_args * sizeof(kwarg));
    int i;
    for (i = 0; i < num_args - 1; i++) {
        head[i] = (kwarg){.name=arg_names[i+1], .count=0};
    }
    return head;
}

int word_counter(char* str, int chars) {
    int i, counter;
    for (i = 0; i < chars; i++) {
        if (isspace(str[i])) counter += 1;
    }
    return counter;
}

char** parse(char* line, int len) {
    char** toks = (char**)malloc(len * sizeof(char*));
    char* item = strtok(line, " ");
    int i = 0;
    //TA of CS303 showed me that strtok was the relevant function
    while (item != NULL) {
        toks[i] = item;
        item = strtok(NULL, " ");
        i += 1;
    }
    return toks;
}

void count(char** tokens, kwarg* table, int row_len, int col_len) {
    int i, j;
    for (i = 0; i < row_len; i++) {
        for (j = 0; j < col_len; j++) {
            if(strcmp(table[i].name, tokens[j]) == 0) {
                table[i].count += 1;
            }
```

```
        }
    }
}

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

//https://stackoverflow.com/questions/3024197/what-does-int-argc-char-
argv-mean/3024202
    //how to use main's arguments
    if (argc < 2) {
        printf("There must be at least one argument.\n");
        exit(-1);
    }

        kwarg* table = init_table(argv, argc);

        char* line=NULL;
        size_t maxlen=0;
        ssize_t n;
        while ((n = getline(&line, &maxlen, stdin)) > 0) {
                int num_words = word_counter(line, (int) n);
                char** tokens = (char**)malloc(sizeof(char*) *
num_words);
                tokens = parse(line, num_words);
                count(tokens, table, argc - 1, num_words);
                free(tokens);
        }

    int i;
    printf("Here is the number of times each keyword appears:\n");
    for (i = 0; i < argc - 1; i++) {
        printf("%s: ", table[i].name);
        printf("%d\n", table[i].count);
    }
        free(line);
        free(table);
        return 0;
}
```