Домашнє завдання №13

Скласти програму (C/C++), яка за допомогою швидкого сортування(з стандартної бібліотеки мови С) дозволяє з вхідного тексту вивести слова, що починаються з заданої літери(решта слів вивести в алфавітному порядку). Сортування потрібно виконати без використання додаткової пам'яті, дозволяється використовувати тільки масив, що зберігає вказівники на початок слів.

Вибір варіанту

Задана літера це перша літера прізвища студента *(записаного латинськими літерами)*

Приклад коду

Наведений зразок коду реалізовує завдання з виконання умови розміщення першими слів, що починаються на літеру 'К'.

Демонстрація виконання завдання з збереженням цілісності вхідного тексту. Вхідний текст є оригінальним текстом.	`//#define METHOD_WITH_FORCE_END_STRING`
Демонстрація виконання завдання без збереження цілісності вхідного тексту. Вхідний текст створюється шляхом копіювання оригінального тексту.	`#define METHOD_WITH_FORCE_END_STRING`

Лістинг

```
#define _CRT_SECURE_NO_WARNINGS
#include "stdio.h"
#include "stdlib.h"
#include <ctype.h>
#include "string.h"

//#define METHOD_WITH_FORCE_END_STRING

#define MAX_WORD_SIZE 64
#define MAX_WORDS 256

#define FIRST_CH 'K'

void scan(char * str, char ** list){
	char s[MAX_WORD_SIZE] = { 0 };
	char * ptr = str;
	char * endPtr = str + strlen(str);
	for (; ptr < endPtr; ++ptr){
		if (sscanf(ptr, "%[^','^'.'^':'^';'^' '^'\t'^'\r'^'\n']", s)) {
			*list++ = ptr;
			ptr += strlen(s);
		}
#ifdef METHOD_WITH_FORCE_END_STRING
		*ptr = '\0';
```

```
#endif
	}
}
void print(char ** list){
#ifdef METHOD_WITH_FORCE_END_STRING
	for (; *list; ++list){
		printf("%s\r\n", *list);
	}
#else
	char s[MAX_WORD_SIZE] = { 0 };
	for (; *list; ++list){
		memset(s, MAX_WORD_SIZE, sizeof(char));
		(void)sscanf(*list, "%[^','^'.'^':'^';'^' '^'\t'^'\r'^'\n']", s);
		printf("%s\r\n", s);
	}
#endif
}

int strcmp__withoutCase(char str1[], char str2[]){
	/*register */int index, str1_tolower, str2_tolower;
	for (index = 0; str1[index] && str2[index]; ++index){
		str1_tolower = tolower(str1[index]);
		str2_tolower = tolower(str2[index]);

		if (str1_tolower != str2_tolower)
		{
			return str1_tolower < str2_tolower ? -1 : 1;
		}
	}

	return 0;
}

int strcmp_K__withoutCase(char str1[], char str2[]){
	/*register */int chr1_toupper, chr2_toupper;
	chr1_toupper = toupper(str1[0]);
	chr2_toupper = toupper(str2[0]);
	if (chr1_toupper == FIRST_CH && chr2_toupper != FIRST_CH){
		return -1;
	}
	else if (chr1_toupper != FIRST_CH && chr2_toupper == FIRST_CH){
		return 1;
	}
	else if (chr1_toupper == FIRST_CH && chr2_toupper == FIRST_CH){
		return strcmp__withoutCase(str1 + 1, str2 + 1);
	}

	return strcmp__withoutCase(str1, str2);
}

int compareFunction(const void* a, const void* b){
	char **arg1 = (char **)a;
	char **arg2 = (char **)b;

	//return (int)strlen(*arg1) - (int)strlen(*arg2); // with case sensitive
	return strcmp__withoutCase(*arg1, *arg2);
}

int compareFunction1(const void* a, const void* b){
	char  ** arg1 = (char **)a;
	char  ** arg2 = (char **)b;

	return strcmp_K__withoutCase(*arg1, *arg2);
}
```

```
void sort(void * data){
	size_t size = 0;
	for (; ((char**)data)[size]; ++size);
	qsort(data, size, sizeof(char*), compareFunction1);
}

int main(){
	char *list[MAX_WORDS] = { NULL };
	char * text = (char*)
		"Sir, in my heart there was a kind of fighting "
		"That would not let me sleep. Methought I lay "
		"Worse than the mutines in the bilboes. Rashly- "
		"And prais'd be rashness for it-let us know "
		"Our indiscretion sometimes serves us well ... "
		; // – Hamlet, Act 5, Scene 2, 4–8

#ifdef METHOD_WITH_FORCE_END_STRING
	char* text_copy = (char*)malloc(MAX_WORDS * MAX_WORD_SIZE * sizeof(char));
	if (!text_copy) {
		return EXIT_FAILURE; // exit(EXIT_FAILURE);
	}
	strncpy(text_copy, text, MAX_WORDS * MAX_WORD_SIZE);
	scan(text_copy, list);
#else
	scan(text, list);
#endif

	sort((void *)list);

	print(list);

#ifdef __linux__
	printf("Press any key to continue . . . \r\n");
	(void)getchar();
#elif defined(_WIN32)
	system("pause");
#else
#endif

	return 0;
}
```