Lab Practice Problem 9

Filename: `practice9_surname.c`

In this activity, you will complete a partially implemented C program that identifies the words appearing in both of the two text files and prints the results to the terminal. This exercise is intended to reinforce your skills in string processing, file input, and function implementation.

Try It Out

Write a program that reads words from two input files, storing up to 200 entries from each, and converts all words to lowercase. After loading both lists, find the words that appear in both files and place them into a new list, while ensuring that there are no duplicates. Finally, sort this list alphabetically and print each word on its own line.

Assumption

You may assume each word is at most 100 characters long, consists only of English letters, and may include at most one internal space (e.g., `Kettle Corn`). No leading or trailing whitespace will appear. When determining matches, treat uppercase and lowercase letters as equivalent.

Your task is to complete the given C program. You can access the starter code here. The following function prototypes are provided that define the functions you need to implement:

`char *trim_string(char *str);`
: A helper function that removes the trailing newline `'\n'` that may be appended when reading input with `fgets()`. Returns a pointer to the same buffer after removing the newline in-place.

`char *to_lower(char *str);`
: Converts all letters in the given string to lowercase. Returns a pointer to the same buffer after modifying it in place.

`int populate_list(char list[MAX_WORDS][MAX_LEN], int *size, int max_size, char *fname);`
: Opens `fname` and reads up to `max_size` words (one per line), converting each to lowercase and storing it in `list`. Writes the number of words read to `size`. Returns `1` on success, or `0` if any error occurs (e.g., the file cannot be opened or read).

`void print_list(char words[MAX_WORDS][MAX_LEN], int size);`
: Prints all stored strings in `words`, one per line.

`int contains_word(char words[MAX_WORDS][MAX_LEN], int size, char *query);`
: Returns `1` if `query` exists in the list; otherwise, returns `0`.

```
void combine_lists(char list1[MAX_WORDS][MAX_LEN], int list1_size,
    char list2[MAX_WORDS][MAX_LEN], int list2_size,
    char list3[MAX_WORDS][MAX_LEN], int *list3_size);
```
: Searches for words that appear in both `list1` and `list2`. Words are added to `list3` in the same order they appear in `list1`. If a word has already been placed in `list3`, it is not added again. `list3_size` is updated to reflect the number of words stored in `list3`.

`void selection_sort(char words[MAX_WORDS][MAX_LEN], int size);`

Sorts the list `words` alphabetically using the selection sort algorithm.

Important You must not modify any other parts of the program. Your task is only to complete the definitions of the seven functions listed above. Do not add or introduce any helper functions. Failure to follow these restrictions will result in point deductions.

Input

The program reads input from two external files, `words1.txt` and `words2.txt`, each containing one word per line. Only the first 200 words from each file will be stored. Each word is at most 100 characters long, consists of English letters, and may include a single internal space (e.g., `"Orange Juice"`), but will not contain leading or trailing whitespace. Words may appear in any combination of uppercase and lowercase letters.

Output

The program outputs all unique words that appear in both input files, converted to lowercase and printed in alphabetical order. Each word is displayed on its own line.

words1.txt

```
apple
banana
Orange Juice
pear
Kettle Corn
kiwi
```

words2.txt

```
grape
KIWI
BANANA
melon
apple
kettle corn
```

Sample Output

```
apple
banana
kettle corn
kiwi
```

Guide Questions

1. Why is it important to convert all words to lowercase before comparing them between files?
2. How does using `contains_word()` help prevent duplicate entries in the final combined list?
3. What could happen if you did not limit the number of words read from each file to `MAX_WORDS`?
4. In `populate_list()`, why must `size` be passed by reference? What would be the consequence of passing it by value instead? Similarly, in `combine_lists()`, why is `list3_size` passed by reference? Could these values be passed by value and still allow the function to communicate the updated size back to the caller? If so, how might the function prototype be redesigned to achieve this? (**Hint:** functions can return values)
5. How is the sorting step implemented in this program, and how does it compare to the previous version that operated on numbers? What changes were necessary to adapt the algorithm to work with strings instead? If the sorting step were removed, how would the output differ? In short, what order the words would appear in.

Sample Solution

The solution will be released after the submission window has closed. You may review it on the activity page at that time.