Programming in C

Deadline: *20 February, 23:59*

Assignment 2b: Wordfinder

Given an m by n grid of letters, (1 <= m, n <= 50), and a list of words, find the location in the grid at which those word can be found. A word matches a straight, uninterrupted line of letters in the grid. A word can match the letters in the grid regardless of case (i.e. upper and lower case letters are to be treated as the same). The matching can be done in any of the eight directions either horizontally, vertically or diagonally through the grid.

Input

Your program should read its input from standard input (`stdin`). The first line gives the size of letter grid as a pair of integers m and n, with 1 <= m, n <= 50. The next m lines contain n letters each; this is the grid of letters in which the words of the list must be found. The letters in the grid may be in upper or lower case. Following the grid of letters, another integer k appears on a line by itself (1 <= k <= 20). The next k lines of input contain the list of words to search for, one word per line. These words may contain upper and lower case letters only (no spaces, hyphens or other non-alphabetic characters).

Output

For each word in the word list, your program should put out a pair of integers representing the location of the corresponding word in the grid. The integers should be separated by a single space where the first integer is the line in the grid where the first letter of the given word can be found, (1 represents the topmost line in the grid, and m represents the bottommost line). The second integer is the column in the grid where the first letter of the given word can be found (1 represents the leftmost column in the grid, and n represents the rightmost column in the grid). If a word can be found more than once in the grid, then the location which is output should correspond to the uppermost occurrence of the word (i.e. the occurrence which places the first letter of the word closest to the top of the grid). If two or more words are uppermost, the output should correspond to the leftmost of these occurrences. All words can be found at least once in the grid.

Example

Sample input:

```
8 11
abcDEFGhigg
hEbkWalDork
FtyAwaldORm
FtsimrLqsrc
byoArBeDeyv
Klcbqwikomk
strEBGadhrb
yUiqlxcnBjf
4
Waldorf
Bambi
Betty
Dagbert
```

Sample Output:

```
2 5
2 3
1 2
7 8
```

Implementation tips

A 2d character array is a natural choice to store the grid of letters. Although not required, it is a good idea write a function to print the grid to verify you have read the input correctly. Because of the limits on the grid size and word list you can use static arrays to store the input data.

Try to avoid magic numbers and literal constants. Instead use definitions such as:

```
/* Fixed size limits for all data structures. */
#define BUFSIZE 512
#define GSIZE 50

/* grid_t: 2d char array to store our letter grid. */
typedef char grid_t[GSIZE][GSIZE];
// ..
```

or try to use constants where possible:

```
/* Fixed size limits for all data structures. */
const int bufsize = 512

/* Cannot define array size with C "constants", use define. */
#define GSIZE 50

/* grid_t: 2d char array to store our letter grid. */
typedef char grid_t[GSIZE][GSIZE];
// ..
```

Bonus Assignments

Online judge

This assignment is taken from a set of practice exercises that are used to train for programming competitions. This means you can check your implementation online with the Uva online judge system [1]. The assignment ID is 10010 and the name is "Where's Waldorf?". You need register to be able to view and submit assignments. Note that the programming challenge version is slightly different. It requires ANSI C and you need to solve multiple lettergrids. See the description on the Uva online judge system [1] for the details. As a bonus exercise you can extend your wordfinder such that it is accepted by the online judge.

Performance

Can you find ways to make your program more efficient? What are the ways to reduce the number of comparisons that you need to perform to find all the words. Implement your ideas and test if your new approach is faster.

Visual improvements

In addition to the coordinates of the words try to emphasize the words in the letter grid using colors or in some other way. Experiment and see what works best to make the different words stand out.

1(1, 2) http://uva.onlinejudge.org