Implementation of reverse command

In this task we implement reverse command: the content of files are printed to the standard output with line numbers in reverse order and mirrored. For example if the content of test.txt is:

apple

peach

plum

…then the result of reverse command is:

3 mulp

2 hcaep

1 elppa

We may assume that the content of any file fits in the memory. (Note: the base of this task is rev Unix utility which prints the content of inputs in original order without line numbers and but with mirrored lines.)

Base task (9 points)

Define a preprocessor symbol named INITCAP which is used in the initialization of a dynamic array that contains the lines of the input file. The value of this macro token should be 8. Also define another preprocessor symbol named BUFSIZE which is used as the maximum length of lines. We may assume that the lines are not longer than 1024 characters.

In the first solution the program can read one file of which the content is printed as described above. The filename is given as a command line argument in the main program. If the user doesn't provide any arguments then an error message should be printed and execution should stop. The behavior of our program is the same if the file can't be opened.

Create a function named read which reads the lines of the file. It gets a file pointer (fp), an array of strings stored in heap (lines) and an integer pointer (lSize) as parameter. The function returns the array. Initial size of lines array is INITCAP. Read the content of the file by lines and store them in lines. When the number of lines reaches the array's capacity then its size should be double. For example if the array size is 8 in the beginning then it should be resized to 16. The number of array elements should be returned through lSize pointer (which is not necessarily the same as the array size).

Create a write function which prints the strings in reverse order, mirrored and with line numbers. This function gets the string array and the array size as parameter.

Create a reverse function which gets a file pointer as parameter. Define a dynamic string array with the initial size. Create an integer variable which will contain the number of lines. Use read and write functions for reading and writing lines.

The main program calls reverse function after opening the file for processing it.

Modularity (3 points) Separate the program to translation units. Function definitions should go to separate translation units to which a header file belongs. Don't forget header guards.

Processing multiple files (2 points) Redesign the program so it is able processing multiple files. For example if test.txt is given twice then the output is this:

3 mulp

2 hcaep

1 elppa

3 mulp

2 hcaep

1 elppa

In this case, the main program performs the operations for each command-line argument. If processing of any parameter fails then the execution should continue with the next parameter.

Reading from standard input (1 point) Redesign the main program so it reads the content of standard input if no command line arguments are given. The lines in this case should arrive from the keyboard and the output is printed accordingly.

(Hint 1: File pointer is a stream which can be substituted with standard input (stdin).) (Hint 2: EOF is a special character which indicates the end of the file. This can be entered by Ctrl-D in Linux and by Ctrl-Z in Windows.) Suggestions Don't forget checking if dynamic memory allocation succeeded. Also don't forget the deallocation of dynamic memory. If you're using fgets then don't forget that a line can be shorter than the maximum buffer size. This function also reads new line characters.