Assignment 3: Simple vi

This assignment asks you to use the demo program `simplevi.c` as a starting point to implement additional features and fix some of its current flaws. The `simplevi.c` code, located on the server in the directory `/home/class_stuff/unix_demos/chapter05`, is a very rudimentary text editor designed to emulate an extremely simple version of `vi`. In its current form, it only allows the user to insert text; it has no means of deleting text or even backspacing. It does not allow the user to open an existing file, allowing the user instead to create text by typing into it from the terminal's keyboard. It does allow the user to save the entered text, but only to a file whose name is hard-coded into the program. In addition, in its current form, there is a fixed limit on how large the file may be and how large lines may be. Lastly, it has not yet implemented vertical scrolling (although that may be fixed shortly).

The implementation itself is not designed to be efficient. It uses a fixed size linear array of characters to represent the text buffer, and each insertion requires shifting the entire array. The same is true when new lines are added, as there is an array of line lengths that keeps track of the lengths of all text lines.

You are to remedy the above deficiencies as follows:

1. Allow the user to enter the name of a file on the command line and fill the text buffer with its contents if one is provided.
2. In command mode, the user should be able to enter the `x` command to delete a single character under the cursor.
3. In command mode, the user should be able to enter the `dd` command to delete the current line.
4. In input mode, when the user enters the backspace key, the character to the left of the cursor should be removed from the screen and from the buffer.
5. In last line mode, the allowed commands should be "`q`", "`w`", "`wq`", "`w filename`", and "`q!`". They should have the same meanings as they do in `vi` – "`q`" quits without saving, "`w`" saves to the file that was opened, "`w filename`" to the file `filename` (without changing the currently opened file, unlike apps such as *MS Office* and *OpenOffice*), and "`q!`" quits without saving a modified file. Thus, the program must detect whether a save took place since the last modification of any kind. For this purpose, a modification is defined as entering input mode whether or not anything was done, and any form of deletion.
6. The program should gracefully handle when the text gets larger than the allocated buffer, either by reallocating a larger one, or letting the user save and quit.
7. If vertical scrolling has not been implemented in the demo by the time you start it, then add it to it.
8. Optional: the arrow navigation keys do not use `vi`'s model of navigation. In `vi`, the up and down arrows move the cursor to the offset within the text line of the last offset of any horizontal movement, or to the end of the line if the offset is greater than the line's length. In `simplevi`, the cursor is moved to the offset of the current line. In other words, it does not preserve the offset from last horizontal movements. Make `simplevi` behave like `vi` in this respect.

EXIT STATUS

`0`	If it succeeded.
`1`	If it crashed for any reason.

Submitting the Assignment

Create a directory in

```
cs82010.gc.cuny.edu:/home/class_stuff/cs82010/projects/project3
```

named `username_hwk3`. Give it permissions 700 so that only you have access to it. Within that directory, put all project-related files. For example, if you have a multi-file project with a Makefile, all files should be there. If you use a Makefile, make sure you name the executable `simplevi` when it is built. If you write a multi-file program, you must create a Makefile, so that I can compile and build it with the single command "make." If it is a single file, name it `simplevi.X`, where `X` is the GNU extension for the language (.c for C, .cpp or .C for C++, etc.)

The program must be well-documented and must contain a preamble or prologue at the very top like the ones in my demo programs, which provides authorship, use, and other pertinent information. All functions should have documentation that details what their parameters are and what its return value is.