The final project is a text editor that operates on LINES of text (no text wrapping from line to line), so it is like a PROGRAM-CODE editor. It can be either an MFC or a WPF project, although I recommend using MFC.

Each line has a proper linend (CR+LF which is \r\n) as required on a Windows system, which does NOT display, nor is it accessible by the user. The window initially shows 10 lines of data, but can be re-sized by the user to show more.

See below for the requirements for commandline commands, suffix commands and other aspects of the program.

At the bottom of this list of requirements, there’s an EXAMPLE of what the final project could look like, depending on YOUR implementation and the actual requirements. This EXAMPLE is based on a program named Kedit, which was based on the IBM program called Xedit (still in use).

Here are the actual requirements for your program:

It has the following rows at the top, in this order:

* The name of the file being edited, the “current” line # (the line at the top of the view), # lines of the file
* A command (menu) bar. All commands listed in the table below MUST have a corresponding menu item if they are **boldfaced** in the table. Commands may be done via a dropdown menu or by being typed on the commandline (see below). There MUST be a “Help” menu item
* a toolbar with any icons you need
* a commandline (denoted by the ====> on a red background in the example)
* the file editing area which contains:
  + suffix column (denoted by the = = = = = rows)
  + a top of file/end of file indicator line
  + a split view indicator line (when the screen is split). Splitting the view is 10 pts extra credit! A “split” command must be available to make it work. The split areas are separately scrollable, but data can be copied/moved between them.
  + a hidden rows indicator line (when rows are hidden)

There are more indicators at the bottom (see the example) below the view area containing:

* the line # the cursor is on
* the column # the cursor is on
* the size of the file
* the count of changes (not required for the assignment)
* file size
* # files being edited in the “ring” of files

Grading:

You start with a 100.

* Any missing or deficient rows at the top costs 20 points each
* A missing or non-working command line costs 100 points
* For each missing command or command that does not work, you lose 10 points.
* For each suffix command that is missing or does not work you lose 5 points

|  |  |
| --- | --- |
| Commandline/menu commands  In all the these descriptions “#” indicates a number the user can type | Explanation (NOTE: unless prohibited below, the first letter of a command can be used instead of the entire command. I.e.; “d” instead of “Down” |
| **Save** | Must be spelled out fully. No parameters. Overwrites the original file. If no file exists, prompt for a path and name. |
| **Save as** | Must be spelled out fully. Saves the file to the name specified on the commandline |
| Open (menu command only) | Opens the specified file reads it into memory, closes it unchanged, but ready for editing |
| Search | (NOT a popup window) Allows the user to search for a string. See rules below these tables for how a user can specify the search. |
| # | skip to line # (from the top) of the file |
| Up # (like moving a slider up) | Scroll up # lines from the current display (display more data toward the top of file) |
| Down # (like moving a slider down) | Scroll down # lines from the current display (display more data toward the end of file) |
| Left # (like moving a slider left) | Scroll left # columns (the view shows more data on the left if any exists) |
| Right # (like moving a slider right) | Scrolls to the right # columns (shows more data on the right if any exists) |
| Forward | Scrolls one “screenfull” toward the end of the file |
| Back | Scrolls one “screenfull” toward the top of the file |
| Setcl # | Must be spelled out fully. Defines which line number (of the lines on the screen, NOT the lines of the file) is the “current line” until further notice. The “current line” is displayed in a red font. The default “current line” is the 1st line of data on the screen. |
| Change | (NOT a popup window) Finds & modifies a searched-for string, starting with the defined “current line”  e.g.; c/abc/ABC/n1 n2  changes abc to ABC on n1 lines n2 times per line. Implementing n2 is optional. |
| Help | Opens a window with a list of the above commands and their usage |

When scrolling Forward or Back, the amount of scrolling is ONE LINE less than the total # of lines currently being displayed. E.g.; if currently displaying 20 lines, then “Forward” scrolls 19 lines toward the EOF. This leaves the previous last-visible-line as the new first-visible-line.

|  |  |
| --- | --- |
| Suffix commands | Explanation |
| i# | Insert |
| x# | Exclude (hide) # lines (default #=1) both from view and from actions – show a “hidden line” marker across the screen with its own suffix area for suffix commands. |
| s# | Show # lines (default n=1) of those excluded. The lines to be shown are the LAST # lines of the # hidden.  (must be entered on the “hidden lines” indicator line) |
| a | Insert AFTER this line |
| b | Insert BEFORE this line |
| c# | Copy # lines (default #=1) starting at this line |
| m# | Move # lines (default #=1) starting at this line |
| “ | Duplicate this 1 line |
|  |  |

## Specifying a search

This must be implemented as both a command-line command and with a dialog box, so the user can do it either way.

Command-line example: find /mystring/#lines\_from\_current\_line starting\_column (if >1)

Dialog example:

find what: …..

replace with what:

#lines: (may be an \* meaning form the current line to the end of file

Starting column #:

Do not worry about finding a string with a / in it in the command-line version. This would require a user-defined escape character instead of the /.

## Replacing strings

Similar to a search, but with an extra string in it as follows:

Change /mystring/newstring/lines # of times

(Optional in “Change” - 5 pts extra credit. Must be stated in “help”.) if \* is used instead of an actual number for # of times, then the editor must check for the string EVERYWHERE on each line to be searched. An \* may also be used as the number of lines on which the replacement is to be done, meaning all lines *start at* the “current line”.

