

Explanations for Problem 3.6 of Lab session 5

We are asked to build a simple text editor. We will create a program which is a simple command line text editor (i.e. no windows, mouse or screen positioning).

The limitations of this editor are:

- Can cope with files of up to 3000 lines of at most 300 characters per line (see symbolic constants MAX_LINE and MAX_LINE_LEN defined in ted.h)
- All commands are one letter commands followed by optional parameters
- It supports the following commands:
 - o **s** – set the line for current operation
 - o **h** - show help on commands
 - o **i** - insert user provided text before current line; end input with Ctrl/Z
 - o **a** - append user provided text after current line; end input with Ctrl/Z
 - o Delete operations:
 - **d** with no parameters deletes current line
 - **d** *lineNo* : delete line *lineNo*
 - **d** *from, to* : delete all lines in interval [*from*, *to*]
 - o Copy operations
 - **c** with no parameters copies current line to scrap
 - **c** *lineNo* : copy line *lineNo*
 - o **c** *from, to* : copy all lines in interval [*from*, *to*]
 - o **p** - paste : pastes scrap buffer before current line
 - o Cut operations
 - **k** with no parameters cuts current line to scrap
 - **k** *lineNo* : cut line *lineNo*
 - **k** *from, to* : cut all lines in interval [*from*, *to*]
 - o View operations
 - **v** with no parameters lists current line
 - **v** *lineNo* : list line *lineNo*
 - **v** *from, to* : list all lines in interval [*from*, *to*]
 - o **n** *filename* – name file to which to write editor buffer contents
 - o Load operations
 - **l** with no parameters loads file whose name was given with the **n** command
 - **l** *fileName* - load *fileName* into editor
 - o Write operations
 - **w** with no parameters writes editor buffer to the file whose name was given with the **n** command
 - **w** *fileName* - write editor buffer to *fileName*. If file exists it is overwritten
 - o **q** - quit without saving changes
 - o **e** - exit and write changes to named file

To achieve this functionality we will design a program composed of three modules:

- A module containing the execution starting point – i.e. the function main
- A dispatcher module, called **dispatch**, which accepts commands from the user
- A support module, called **ted**, containing the functions which implement editor functionality

The program is actually an infinite loop in the dispatcher, waiting for user commands. When a command is accepted, the appropriate function in the support module is called to execute that command.

The implementation uses three string arrays: lines, scrap, and buffer.

- The array **lines** contains the text to be edited.
- The array **scrap** is used as scrap area for commands *copy*, *cut* and *paste*.
- The array **buffer** is used for getting text from the user for commands *insert* and *append*.

There are two auxiliary functions, *moveUp* and *moveDown*. *moveUp* moves a block of text to make room for text to be inserted. *moveDown* copies over deleted text the rest of lines array overwriting the deleted area. Note that *cut* is implemented by an invocation of *copy* and then of *delete*.