Answers to Even-Numbered Exercises 11

from page 536

1. Given a buffer full of English text, answer the following questions:
   a. How would you change every instance of `his` to `hers`?
   b. How would you do this only in the final paragraph?
   c. Is there a way to look at every usage in context before changing it?
   d. How would you deal with the possibility that `His` might begin a sentence?

2. What command moves the cursor to the end of the current paragraph? Can you use this command to skip through the buffer in one-paragraph steps?

   The META-} command moves the cursor to the end of the current paragraph; you can use it repeatedly to move through the buffer by paragraphs.

3. Suppose that you get lost in the middle of typing a long sentence.
   a. Is there an easy way to kill the botched sentence and start over?
   b. What if only one word is incorrect? Is there an alternative to backspacing one letter at a time?

4. After you have been working on a paragraph for a while, most likely some lines will have become too short and others too long. Is there a command to "neaten up" the paragraph without rebreaking all the lines by hand?

   Give the command META-x **mark-paragraph** (or META-h) while the cursor is on the paragraph you want to reformat, and then give the command META-x **fill-region**.

5. Is there a way to change the whole buffer to capital letters? Can you think of a way to change just one paragraph?

6. How would you reverse the order of two paragraphs?

   With the cursor on the first paragraph, use META-h to define the paragraph as Region, Kill Region with META-x **kill-paragraph**, use META-} to move the cursor to the end of the second paragraph, and use CONTROL-Y to yank the killed paragraph.

7. How would you reverse two words?

8. Imagine that you saw a Usenet posting with something particularly funny in it and saved the posting to a file. How would you incorporate this file into your own buffer? What if you wanted only a couple of paragraphs? How would you add > to the beginning of each included line?

   To read a file into the current buffer, move the cursor to where you want to add the file, and give the command CONTROL-X **i** (insert-file).

   There are several ways to add only a few paragraphs from a file: You can read the entire file and delete the parts you do not want; or you can read the file into its own buffer with CONTROL-X CONTROL-F, kill the part you want, go back to the original buffer with CONTROL-X **b** (switch-to-buffer), and yank the killed text with CONTROL-Y.

   To place a greater-than sign followed by a SPACE at the start of each line of the new text, place the cursor at the beginning of the new text, give the command META-x query-replace-regexp ^RETURN >SPACE RETURN, and respond with SPACE to each prompt until you get to the end of the new text; then respond with RETURN.

9. On the keyboard alone emacs has always offered a full set of editing possibilities. For any editing task there are generally several techniques that will accomplish the same goal. In the X environment the choice is enlarged still further with a new group of mouse-oriented visual alternatives. From these options you must select the way that you like to solve a given editing puzzle best.

Consider this Shakespearean fragment:

```
1. Full fathom five thy father lies;
2.   Of his bones are coral made;
3. Those are pearls that were his eyes:
4.   Nothing of him that doth fade,
5. But doth suffer a sea-change
6. Into something rich and strange.
7. Sea-nymphs hourly ring his knell:
8.           Ding-dong.
9. Hark! now I hear them--
10.      Ding-dong, bell!
```

The following fragment has been typed with some errors:

```
1. Full fathiom five tyy father lies;
2. These are pearls that were his eyes:
3.   Of his bones are coral made;
4.   Nothin of him that doth fade,
5. But doth susffer a sea-change
6. Into something rich and strange.
7. Sea-nymphs hourly ring his knell:
8.           Ding=dong.
9. Hard! now I hear them--
10.      Ding-dong, bell!
```

Use only the keyboard to answer the following:

a. How many ways can you think of to move the cursor to the spelling errors?

b. Once the cursor is on or near the errors, how many ways can you think of to fix them?

c. Are there ways to fix errors without explicitly navigating to/searching for them? How many can you think of?

d. Lines 2 and 3 are transposed. How many ways can you think of to correct this situation?

Running xemacs (give the command **emacs** from an xterm or another terminal emulator window), use the mouse to answer the following:

e. How do you navigate the cursor to a spelling error?

f. Once the cursor is on or near the errors, how many ways can you think of to fix them?

g. Lines 2 and 3 are transposed. Is there a visually oriented way to fix them?

h. Is there a visual way to correct multiple errors (similar to META-%)?

Advanced Exercises

10. Assume that your buffer contains the C code shown here, with the Major mode set for C and the cursor positioned at the end of the **while** line as shown by the black square:

```
/*
* Copy string s2 to s1.  s1 must be large enough
* return s1
*/
char *
strcpy(s1, s2)
register char *s1, *s2;
{
     register char *os1;

     os1 = s1;
     while (*s1++ = *s2++)
     ;
return(os1);
}

/* Copy source into dest, stopping after '\0' is copied, and
   return a pointer to the '\0' at the end of dest.  Then our caller
   can concatenate to the dest string without another strlen call. */
char *
stpcpy (dest, source)
     char *dest;
     char *source;
{
  while ((*dest++ = *source++) != '\0') ■
     ; /* void loop body */
  return (dest - 1);
}
```

a. What command moves the cursor to the opening brace of **strcpy**? What command moves the cursor past the closing brace? Can you use these commands to skip through the buffer in one-procedure steps?

With the cursor at the start of the file, give the command CONTROL-META-e to move the cursor past the closing brace of **strcpy**; CONTROL-META-a moves it back to the opening brace. You can use these commands to skip from procedure to procedure.

b. Assume the cursor is just past the closing parenthesis of the **while** condition. How do you move to the matching opening parenthesis? How do you move back to the matching close parenthesis again? Does the same command set work for matched [] and {}? How does this differ from the vi % command?

CONTROL-META-b moves backward over an expression and CONTROL-META-f moves forward over an expression. The expression can be delimited by (), [], or {}.

The vi % command requires that you position the cursor on the same line as, and on or to the left of, the closing element of the expression. Then % jumps between the opening and closing elements.

c. One procedure is indented in the Berkeley indention style; the other is indented in the GNU style. What command reindents a line in accordance with the current indention style you have set up? How would you reindent an entire procedure?

Enter TAB while the cursor is positioned anywhere on a line to reindent the line to the current indention style. Position the cursor before a pair of matched braces and enter CONTROL-META-q to reindent the lines within the braces to the current style.

d. Suppose that you want to write five string procedures and intend to use **strcpy** as a starting point for further editing. How would you make five duplicate copies of the **strcpy** procedure?

Move the cursor to the beginning of the word `strcpy` and enter CONTROL-SPACE to set mark. Move the cursor to the line past the closing brace and enter META-w to copy the region nondestructively to the Kill Ring. Finally, enter CONTROL-Y five times to yank five copies of the killed region into the work buffer.

e. How would you compile the code without leaving emacs?

After saving the buffer, give the command META-x **compile**. You will be prompted for a command; respond with the command to compile the file you are working on. The output of the compilation appears in a buffer named ***compilation***.