```
/**********************************************************
 * read_word: Reads the next word from the input and      *
 *            stores it in word. Makes word empty if no   *
 *            word could be read because of end-of-file.  *
 *            Truncates the word if its length exceeds    *
 *            len. Returns the number of characters       *
 *            stored.                                     *
 **********************************************************/
int read_word(char *word, int len);
```

Of course, we're careful to change the comment that accompanies `read_word`. Next, we change the definition of `read_word` in `word.c`:

```
int read_word(char *word, int len)
{
  int ch, pos = 0;

  while ((ch = read_char()) == ' ')
    ;
  while (ch != ' ' && ch != EOF) {
    if (pos < len)
      word[pos++] = ch;
    ch = read_char();
  }
  word[pos] = '\0';
  return pos;
}
```

Finally, we modify `justify.c` by removing the include of `<string.h>` and changing `main` as follows:

```
int main(void)
{
  char word[MAX_WORD_LEN+2];
  int word_len;

  clear_line();
  for (;;) {
    word_len = read_word(word, MAX_WORD_LEN+1);
    if (word_len == 0) {
      flush_line();
      return 0;
    }
    if (word_len > MAX_WORD_LEN)
      word[MAX_WORD_LEN] = '*';
    if (word_len + 1 > space_remaining()) {
      write_line();
      clear_line();
    }
    add_word(word);
  }
}
```