Pseudocode Basics

There are two basic rules of pseudocode. Pseudocode should

1. Specify the solution to the problem adequately
2. Use structured English in place of C++ whenever doing so aids readability and brevity

Of course the devil is in the details. In this course, we define adequately as follows:

> Pseudocode is adequate if one of your peers could implement your pseudocode without having to make any significant implementation decisions.

Thus, if you are asked to provide pseudocode for finding the length of the longest all-uppercase word sequence in a line, it probably is not acceptable to write

> `longest` = length of the longest all-uppercase word sequence in `line[]` by scanning it

even if *you* think it is obvious how such a "scanning" operation should be implemented, because the necessary code is not likely to be immediately obvious to all of your peers. Instead, you should write something like

```
longest = find length of the longest all-uppercase word sequence in line[]:
    longest = 0
    while not at the end of the line
        if we're at the start of an uppercase word sequence
            start = current position in line[]
            scan forward until we reach a character that isn't an upper-case letter or a space
            back up until we hit an uppercase letter
            end = current position in line[]
            length = end - start + 1;
            longest = length if length is greater
        advance current position by one character
```

There are several things to note about the preceding code:

- No C++ compiler would compile the code as it stands
- Structure is shown using indentation (there are no braces)
- A colon is used to introduce a more detailed implementation description
- Some lines are actually valid C++ (e.g., "`length = end - start + 1;`")
- Simple variables are not declared

- Some variables are not explicitly named (e.g. "current position in `line[]`") and are sometimes used implicitly (e.g. "not at the end of the line" uses both the "current position" variable and the "`line[]`" variable)
- Some *very simple* loops are implicit (e.g., "scan forward until…" requires a loop in the implementation)