CS 1132 lecture 8

I. Character arrays
   a. Character: a single letter, digit, punctuation symbol, space, etc.
   b. Denote (sequences of) characters in Matlab by enclosing in single quotes
      i. Beware copy-paste from Office programs – fancy quotes don't work
   c. May have multiple dimensions (like numeric arrays); still need to be rectangular
   d. Character array contents vs. code
   e. Often called "strings," but differentiate from Matlab's new `string` type, which corresponds to characters enclosed in double quotes
      i. `string` not covered in 1132: less in common with other Matlab concepts, can use cell arrays of char vectors instead

II. Type `char`
   a. Represents a single character
   b. A primitive built-in type in Matlab, along with `double` and `logical`
   c. Takes up a fixed amount of memory (16 bites = 2 bytes)
      i. Built-in command `whos` shows memory usage
      ii. Allows constant-time array indexing

III. Syntax
   a. Multiple chars in single quotes automatically concatenates
   b. Can concatenate multiple char vectors with brackets
   c. Use array slicing notation to get substrinsg

IV. Example: remoteChar()
   a. Conditional accumulation pattern
   b. Can implement with a 1-liner using logical indexing

V. Functions
   a. isletter(), isspace()
   b. lower(), upper()
   c. strcmp(): prefer to vectorized ==

VI. Encoding
   a. Every character is associated with a number ("codepoint")
      i. Get codepoint by converting to double with `double()` function

            ii. Convert codepoint to char with `char()` function
    b. ASCII: 128 characters, including Latin letters, English punctuation, Arabic digits, and control sequences
    c. Unicode: nearly 150,000 characters from most languages; ASCII is a subset
        i. Most common codepoints can be represented with 16 bits, so Matlab uses UTF-16 encoding
    d. Digits are adjacent and in order, capital letters are adjacent and in order, lower-case letters are adjacent and in order
        i. Allows meaningful addition/subtraction (yields double result), comparison
    e. Reminder: character vectors are vectors, support vectorized arithmetic

VII. Example: toUpper()
    a. Decide whether character is a lower-case Latin letter
    b. Shift to corresponding upper-case letter
    c. Extension: vectorize

VIII. Example: replace words
    a. Construct replacement word
    b. Iterate 1D stencil (moving window); careful with loop bounds
    c. String comparison: use strcmp()