CS 1132: Lecture 10 (cell arrays, file I/O)

I. Primitive arrays
   a. Homogeneous data type
   b. Rectangular
      i. Poor fit for lists of strings

II. Cells
   a. Value may have any type (including nested cell) - heterogeneous
   b. Value may be an array (including nested cell array) – nested
      i. Good for lists of strings
   c. A cell array itself _is | is not_ still rectangular
   d. Syntax:
      i. Curly braces for creation, nesting
      ii. Curly braces for indexing
      iii. Square brackets for primitive arrays, concatenation

III. Example: Roman numerals
   a. Still a decimal system, but each "digit" may be written with 0-4 characters
   b. To translate from Arabic to Roman, construct a lookup table for each decimal place
      i. Each entry in table may have a different length
   c. If thousands place stops at MMM, can't use 2D cell array, but can use nested cell arrays (or 4 separate named arrays)
   d. Nesting order needs to increment ones place the fastest
      i. Outermost loop should be ____________ place

ii. Innermost loop should be ____________ place

IV. File input/output

  a. Needed to process non-trivial data from real world

  b. Needed to move data between different systems

  c. 3-step process:

    i. Open file: fopen()

      1. Input is filename; return value is "file identifier" (used as argument for all other file I/O functions)

      2. Need to specify if opening for **r**eading, (re)**w**riting (creates or truncates), or **a**ppending

        a. Careful – don't accidentally overwrite important files!

    ii. Read from or write to file

      1. To write text, use fprintf() with fid as first arg (to print strings, use '%s' format specifier)

    iii. Close file (don't forget this step!): fclose()

      1. Analogous to end keyword, but Matlab can't catch if you omit it

      2. Returns a status code, so end line with semicolon

  d. Files know how large they are and track how much you have read, so you can ask if you're at the end: feof()

V. Line-oriented text files

  a. For random access, read whole file into memory, store each line in a cell array element

b. Lines are separated with newline characters (one or two, depending on OS): ‘\r’, ‘\n’
c. fgetl(): read a line, discard newline characters at end
d. str2double(): convert text numbers (char array) to numeric values (double)
e. If random access to lines is not required, can process one line at a time as a “stream”
    i. Only store the data you’re interested in (NORAD example)