Problem 1. (20 points) \texttt{vals.icn}

In this problem you are to write a program named \texttt{vals} that performs simple calculations on the numbers, if any, contained in blocks of text indicated by the user.

\texttt{vals} operates on a text file that is named on the command line. As an example, consider a file named \texttt{xmlsizes} that happens to contain some \texttt{wc} output:

\begin{verbatim}
% cat xmlsizes
  25  35  622  books.xml
  34  132 1352  cook.xml
  542  766 22034  garment.xml
   14   22  306   months.xml
    8   21  153   quotes.xml
  327  327  8050  rptcard.xml
   10   25  207   text.xml
  960  1328 32724  total
%
\end{verbatim}

The command "\texttt{vals xmlsizes}" produces this window:

\begin{center}
\includegraphics[width=0.5\textwidth]{fig1.png}
\end{center}

By default, the text is displayed in a 16-point typewriter font. The window is precisely sized based on the longest line and the number of lines.

The default point size can be overridden with the \texttt{-s} option:

\begin{verbatim}
vals -s 12 xmlsizes
\end{verbatim}

If present, \texttt{-s size} precedes the filename.
Using a left button click and drag the user sweeps out a rectangle of text that is of interest:

The rectangle is rubberbanded and "snaps" to the character "cells". When the left button is released, a dialog box appears:

The dialog box shows that seven lines are in the selected region and that all seven lines contain a numeric value. The minimum, maximum, sum, and average are displayed as well. Note that minimal formatting is done—there is no horizontal alignment and the value displayed for the average is simply what Icon produces for \( \text{string}(960/7.0) \).

The user may select as little as one character or the full window contents.

The dialog box is dismissed by clicking the Okay button. The rubberbanded rectangle then disappears and the user may make another selection.

Typing a "q" exits vals.
Here is another example (vals vals.1):

In this case four lines are selected but the text in two of them does not represent a numeric value. The values for minimum, et al. are computed based on only two values (1 and 2).

Note that because the file vals.1 ends with a blank line there is a blank line at the bottom of the window. It also happens that one line has a space character following the rightmost digit and that produces the blank column on the right edge of the screen.
Here is a third example: `(vals times)

Neither line in the left hand example's selection has a numeric value; all that is reported is that there are two lines in the selection.

In the right hand example the values contain a decimal point and therefore the calculated values are shown as real numbers.

Here are some good ways to simplify the problem:

Assume that the user will drag out the rectangle with an upper-left to lower-right motion.

Assume that there are no tab characters in the input.

Don't worry about checking command line arguments for validity.

Don't worry about the possibility of there being insufficient screen space to display all the text. Assume it will fit.

Don't worry about precisely matching the behavior of the reference version when
sweeping out the selection. All that's important is that the selection box grow (or shrink) in increments of the character width and/or height and that there's no ambiguity about which characters are selected.

Don't worry about the case of dragging the pointer outside of the window and then releasing the button.

**Implementation Notes**

To determine if the text of a line of the selection represents a value, call the `numeric()` function with that text. If `numeric()` succeeds, it is a value. By this rule text such as "x 10", "3  7", and "7 x" does NOT represent a value. Use the value produced by `numeric()` in the subsequent computations.

Use the `Notice(s1, s2, ..., sn)` procedure to produce the dialog box. Here is a trivial program that uses `Notice()`:

```plaintext
link graphics, split
procedure main()
    WOpen("rows=30","cols=60")
    while line := WRead() do
        Notice!split(line)
end
```

When the dialog is dismissed with the Okay button, `Notice()` returns.

Hint: Avoid tricky failure situations by using `left()` to pad all input lines to the length of the longest line.

**Problem 2. (10 points) amuse.icn**

In this problem you are to write a small program of your design that does something "interesting". It may use random numbers, text, color, event handling, or whatever you wish. A program with complexity similar to the examples on slides 26-28, 30-31, 34, or 38 would be entirely adequate.

If you're short on ideas for a problem, or if you want to be sure that your idea is suitable, ask the instructor.

Along the way to an imagined program you may find a buggy version that has interesting behavior. If the results are interesting don't hesitate to call a bug a feature and declare it done.

Be sure to include header comments that describe how to exercise the program.

**Miscellaneous**

Aside from code appearing in the class handouts, in the texts, or in messages from the instructor you may not seek, study, refer to, use, incorporate, etc., any existing solutions or portions thereof for any problem.
Reference Version

Both UNIX and Windows reference versions of vals are in /home/cs451/a8, along with some data files. Notify the instructor if the behavior of a reference version seems to contradict or extend a problem specification.

Deliverables

Use turnin with the tag 451_8 to submit your solutions for grading. The deliverables for this assignment are vals.icn and amuse.icn. If your amuse.icn needs data files, submit those as well.