Problem 1. (10 points) longest.icn

Write an Icon program that reads lines from standard input and upon end of file, writes the longest line to standard output. If there are ties for the longest line, longest writes out all the lines that tie. If there is no input, longest should produce no output.

Examples: (note that % is the shell prompt)

```
% cat lg.1
  a test
  for
  the program
  here
% longest < lg.1
  the program
% cat lg.2
  xx
  a
  yy
  b
  zz
% longest < lg.2
  xx
  yy
  zz
% longest < /usr/share/lib/dict/words
  electroencephalography
% grep ^c /usr/share/lib/dict/words | longest
  contradistinction
  contradistinguish
  counterproductive
% cat lg.3
  (empty file)
% longest < lg.3
%
```

Restrictions: No comparison operators (shown on slides 33-35) may be used. No arithmetic operations, such as addition and subtraction, may be used. The only datatypes you may use are integer and string. In particular, you may not use lists.
Problem 2. (15 points) seqwords.icn

For this problem you are to write a program that reads a series of words from standard input, one per line, and then prints lines with the words sequenced according to a series of specifications, also one per line and read from standard input.

Here is an example with seven words and three specifications:

```
% cat sw.1
one
two
three
four
five
six
seven.
1
2
3.
3
2
1
1
2
3.
%
% seqwords < sw.1
one two three
two three one one two three
seven%
```

Note that lines containing only a period (.) end the word list and separate the specifications. For output, words are separated by blanks.

Assume that there will always be at least one word and at least one sequencing specification. Assume that there will be at least one number in each sequencing specification and that all numbers in the sequencing specifications are valid. Assume that words are at most 1000 characters in length.

Restrictions: The only datatypes you may use are integer and string. In particular, you may not use lists.
Problem 3. (20 points) boxlines.icn

Write an Icon program that reads lines on standard input and prints boxes around them, one box per line of input, centering the lines.

Examples:

```
% cat b.1
procedure main()
  write("Hello!)
end
% boxlines < b.1
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>procedure main()</td>
</tr>
<tr>
<td>------------------</td>
</tr>
<tr>
<td>write(&quot;Hello!)</td>
</tr>
<tr>
<td>------------------</td>
</tr>
<tr>
<td>end</td>
</tr>
<tr>
<td>------------------</td>
</tr>
</tbody>
</table>
%
```

Note that the text is centered, ignoring blanks. The size of the boxes is based on the longest line in the file.

```
% cat b.2
  a short line
    a loooooooooooooooooooooooooooong line
       x
          yy
             zzz
% boxlines < b.2
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a short line</td>
</tr>
<tr>
<td>----------------------------------------</td>
</tr>
<tr>
<td>a loooooooooooooooooooooooooooong line</td>
</tr>
<tr>
<td>----------------------------------------</td>
</tr>
<tr>
<td>x</td>
</tr>
<tr>
<td>----------------------------------------</td>
</tr>
<tr>
<td>yy</td>
</tr>
<tr>
<td>----------------------------------------</td>
</tr>
<tr>
<td>zzz</td>
</tr>
<tr>
<td>----------------------------------------</td>
</tr>
</tbody>
</table>
%
```

```
% echo x | boxlines
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
</tr>
<tr>
<td>---</td>
</tr>
</tbody>
</table>
```
% echo x | boxlines | boxlines
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td>-------</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>-------</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>-------</td>
</tr>
</tbody>
</table>
%

Remove both leading and trailing spaces from each input line (see trim() and reverse()). Assume there are no tab characters in the input. Use center() for centering the text. Note that center() produces an imperfect result when, for example, centering a two-character string in a five-character field.

You may assume that there will be at least one line of input.

Restrictions: The only datatypes you may use are integer and string. In particular, you may not use lists. The instructor’s solution employs recursion to accommodate the restrictions.

Reference Versions

Reference versions of these programs and associated data files are available in /home/cs451/a2. The output of your implementations should exactly match the output of the reference versions; use the diff command to compare outputs. If a reference version exhibits a behavior that seems to contradict or extend the specifications for a program, let me know.

Miscellaneous

Note that all problems place restrictions on which language elements may be used. Failure to meet the restrictions may result in a large point deduction.

You are specifically prohibited from directly copying any code, except that presented in class or otherwise provided by me. However, you may study discovered code, such as that found in a textbook—not the code of a classmate—to the point of understanding how it works and then with that knowledge, write your own version.

Each solution should have an overview of a sentence or two, maybe three, describing the basic idea of your approach to the problem.

Deliverables

Use turnin with the tag 451_2 to submit your solutions for grading. The deliverables for this assignment are the files longest.icn, seqwords.icn, and boxlines.icn.