Name:____

CSc 451, Spring 2003
Examination #1
February 25, 2003

READ THIS FIRST

Fill in your name above. Do not turn this page until you are told to begin.

DO NOT use your own paper. If you run out of room, write on the back of the page.

If you have a question that can be safely resolved by making an assumption, simply write down that assumption and proceed. Examples:

> "Assuming `reverse(s)` reverses a string"
> "Assuming `*t` returns the number of keys in table `t`"

If you have a question you wish to ask, raise your hand and the instructor will come to you. DO NOT leave your seat.

You may use all elements of Icon except string scanning and co-expressions (we have not yet covered the latter).

You may use Icon procedures that have appeared on the slides, or been presented in class, or been mentioned in e-mail, or that have appeared on an assignment this semester, or that are mentioned in this exam.

There are no deductions for poor style. Anything that works and meets all problem-specific restrictions will be worth full credit, but try to keep your solutions brief to save time.

You need not include any explanation in an answer if you are confident it is correct. However, if an answer is incorrect, any accompanying explanation may help you earn partial credit.

If you are unsure about the form or operation of a language construct that is central to a problem's solution, you are strongly encouraged to ask the instructor about it. If you're completely puzzled on a problem, please ask for a hint. Try to avoid leaving a problem completely blank—that will certainly earn no credit.

This is a **sixty** minute exam with a total of 100 points and 5 possible points of extra credit. There are 10 regular problems and an extra credit section with 5 problems.

When you have completed the exam, enter your name on the exam sign-out log and then hand your exam to the instructor.

When told to begin, double-check that your name is at the top of this page, and then put your initials in the lower right hand corner of each page, being sure to check that you have all the pages.

Problem 1: (15 points)

Consider a spell-checker word list with entries such as these:

```
abbreviate,s,d,\ing,\ion
bar,s,"ed,"ing
calmest
```

Each entry consists of a word followed by valid suffixes. If a suffix begins with a backslash, it indicates that the last character of the word should be dropped before appending the suffix. If a suffix begins with a double quote character, it indicates that the last character should be doubled before appending the suffix.

Write a program `expand` that reads such a word list and prints all forms of each word. Example, assuming the above lines are in the file `words.dat`:

```
% expand < words.dat
abbreviate
abbreviates
abbreviated
abbreviating
abbreviation
bar
bars
barred
barring
calmest
%
```

Problem 2: (15 points)

Write a procedure `eval(s)` that evaluates string representations of expressions consisting of integer values and the binary operators `+`, `-`, `*`, and `/`. DO NOT worry about precedence—just evaluate in a strict left-to-right order. Assume that the input is well-formed. Use integer arithmetic for all operations. Note that `eval` returns the value of the expression.

```
][ eval("3+5");
   r := 8  (integer)

][ eval("3000");
   r := "3000"  (string)

][ eval("3+50*100");
   r := 5300  (integer)

][ eval("20-3/5");
   r := 3  (integer)

][ eval("1*2+3*4-4/5");
   r := 3  (integer)
```

Problem 3: (10 points)

The built-in function `reverse` reverses only strings. Write a procedure `Reverse(x)` that reverses either strings or lists. If `x` is a list, the reversal is at the top level only. You may use the built-in function `reverse` in your solution.

```
][ Reverse("testing");
   r := "gnitset"  (string)

][ Reverse(["just", "testing"]);
   r := ["testing","just"]  (list)

][ Reverse([[1,2,3], 10, 20, "thirty"]);
   r := ["thirty",20,10,[1,2,3]]  (list)

][ Reverse([[1,2]]);
   r := [[1,2]]  (list)
```

Problem 4: (8 points)

Write a procedure `altbang(s)` that generates the characters of `s` working in from each end in an alternating manner. If `s` is the null string, the result sequence is empty. Examples:

```
][ .every altbang("abcde");
   "a"  (string)
   "e"  (string)
   "b"  (string)
   "d"  (string)
   "c"  (string)

][ .every altbang("1234");
   "1"  (string)
   "4"  (string)
   "2"  (string)
   "3"  (string)

][ every write(altbang("12"));
1
2
Failure
```

Problem 5: (20 points)

Write a program `exttotal` that reads "`ls -s`" output and prints a table of file extensions and the total number of blocks used by files of that type in the current directory. Example:

```
% ls -s
total 879
   1 altbang.icn
 432 extdu.exe
   1 extdu.icn
   4 notes.txt.old
   1 reverse.icn
   2 reverse.u1
   1 reverse.u2
 431 sumex.exe
   1 sumex.icn
   1 toby.icn
   1 unique.icn
   3 x
   0 y
% exttotal
u2             1 blocks
u1             2 blocks
(None)         3 blocks
old            4 blocks
icn            6 blocks
exe          863 blocks
%
```

Files with no extension, such as x and `y`, are grouped under `(None)`. For the file `notes.txt.old`, the extension is considered to be the last element, "`old`". For output, left justify the extensions in a field of width 10 and right justify the block totals in a field of width 6.

Problem 6: (10 points)

Write a program `lensort` that reads a file named on the command line and prints the lines of the file in order of increasing length. The sort does not need to be stable, i.e., lines of equal length need not appear in the same order that they occur in the input file. Don't worry about tabs in the input. If you don't recall the `sortf` function, ask the instructor about it.

Example:

```
% cat txt
here
is
a
test of
this
program
% lensort  txt
a
is
here
this
test of
program
%
```

If the file can't be opened, print an error message and terminate the program by using the `stop()` function.

Problem 7: (1 point each; 5 points total)

Note: See restriction below

Write an expression whose result sequence ...

...is empty:

...is infinite:

...has length 2:

...has length 10:

...has length 100:

RESTRICTION for problem 7 (above!) : Other than literals, you may not use any element of the language more than once. For example, if you use subscripting (string, list or table) for one expression you may not use it in another expression. Alternation and repeated alternation are considered to be two distinct language elements—you may use both.

Problem 8: (2 points each, 8 points total)

Write expressions that have the following result sequences. You may use built-in functions such as `repl(s,n)` but you may not write any helper procedures.

(a) All capital letters in the string `s`. For example, if `s` is "The Right Way", the result sequence would be {"T", "R", "W"}.

(b) The character and position of each character in the string `s`. For example, if `s` is "abc", the result sequence would be {"a", 1, "b", 2, "c", 3} — six values altogether.

(c) The infinite sequence {1, 1, 2, 1, 2, 3, 1, 2, 3, 4, ...}.

(d) The integers in the list `L`, in descending order. For example, if L is ["x", 5, 3, "y", 10, 5, 4.1], the result sequence would be {10, 5, 5, 3}.

Problem 9: (6 points)

Write a procedure `invert(t)` that returns an inverted copy of the table `t` by swapping keys and values. This is best illustrated with examples:

```
][ vtab := table();
   r := T1:[]  (table)

][ vtab["x"] := 10;
   r := 10  (integer)

][ vtab["y"] := 20;
   r := 20  (integer)

][ vtab;
   r := T1:["x"->10,"y"->20]  (table)

][ i := invert(vtab);
   r := T1:[10->"x",20->"y"]  (table)

][ i[10];
   r := "x"  (string)

][ i[20];
   r := "y"  (string)

][ invert(invert(vtab));
   r := T1:["x"->10,"y"->20]  (table)

][ vtab["z"] := 20;
   r := 20  (integer)

][ vtab;
   r := T1:["x"->10,"y"->20,"z"->20]  (table)

][ invert(vtab);
Failure          # Fails because value 20 appears twice
```

`invert(t)` fails if the table `t` contains any values that are not unique.

Assume that the default value for the table is `&null`.

Problem 10: (3 points)

Show the output of this program:

```
procedure main()
    every write(("+"|"*")(2|3, 4|5))
end
```

EXTRA CREDIT SECTION (one point each)

(a) Who was known as "bikmort"?

(b) What is the output of the following expression?

```
every write(every 1 to 10)
```

(c) Write a procedure `defvalue(t)` that returns the default value of table `t`.

(d) List the last names of ten other students in this class. Use of phonetic spelling is acceptable.

(e) Write a good one point extra credit question and answer it correctly.