Name:____ Seat row **and** number: __________

CSc 372, Fall 2001
Icon Examination
October 24, 2001

READ THIS FIRST

Fill in your name and seat row/number above.

Do not turn this page until you are told to begin.

DO NOT use your own paper. Do not write on the opposite side of the paper. If you need extra sheets, ask for them.

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`"

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

You may use all elements of Icon regardless of whether they have been studied in class. You may not use any elements of the Icon Program Library (IPL). (The instructor has said nothing about anything in the IPL during lectures.)

Changed You may use Icon procedures that appear on the slides, or have been presented in class, or have been mentioned in e-mail. Here are some Icon procedures that may be useful: `split`, `atos`, `rev`, and `read_file`

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

On problems that ask you only to write code, you need not include any explanation in your answer if you are confident it is correct. However, if an answer is incorrect, any accompanying explanation may help you earn partial credit.

This is a **forty-five** minute exam with a total of 100 points and eight possible points of extra credit. There are eight regular problems and six extra credit problems.

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

When told to begin, double-check that your name and seat/row are recorded 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)

Write a program `tacdel` that reads lines from standard input and prints the lines in the file in reverse order. If a line contains the character "`@`", the line "`<D>`" appears in place of that line. Example:

```
% cat x
this
is @ some data
here to
test@
with
% tacdel < x
with
<D>
here to
<D>
this
%
```

Problem 2: (15 points)

Write a program `idiff` that examines two files named on the command line and if the files are not identical, prints "`Diffs`". If the files are identical, `idiff` produces no output.

Two files are considered to be identical if and only if they contain the same number of lines and the lines in each are identical and in the same order. Here are four sample input files. Note that `f1` and `f3` are identical.

```
% cat f1
a
test
here
%
```

```
% cat f2
here
is
a
test
%
```

```
% cat f3
a
test
here
%
```

```
% cat f4
1
2
3
%
```

Examples of usage:

```
% idiff f1 f2
Diffs
% idiff f1 f3
% idiff f2 f4
Diffs
% idiff f1 f1
%
```

Problem 3: (15 points)

Write a program `paldate` that searches for palindromic dates between January 1, 2001 and December 31, 2099 and prints those dates. Represent a date such March 15, 2001 like this: `3/15/1`.

Some examples of palindromic dates are `1/1/1`, `1/22/1`, `3/11/3`, and `12/11/21`.

Note that months 4, 6, 9, 11 (April, June, September, and November) have 30 days and assume that February always has 28 days. All the other months have 31 days.

RESTRICTION: You may not use any lists in your program.

The output of `paldate` is always the same:

```
% paldate
1/1/1
1/2/1
1/3/1
1/4/1
1/5/1
1/6/1
1/7/1
1/8/1
1/9/1
1/11/1
...lots more lines...
```

Problem 4: (24 points)

Write a program `total` that reads merchandise descriptions and prices and then computes the total for a list of items to purchase.

Both merchandise descriptions and the list to purchase are read from standard input. One or more lines of merchandise item names and associated prices appear first, one per line. A line containing only a "`.`" signals the end of the descriptions, and that the full names of items to purchase follow. The output of the program is the total price of all the items listed after the end of the descriptions.

Here is a sample input file:

```
% cat total.in
Icon book                    $10
Old lamp with bad switch      $2
Bowling ball                  $1
Perl book                     2c
Pencil                        6c
.
Pencil
Bowling ball
Pencil
Icon book
Old lamp with bad switch
%
```

The program outputs the total cost of the to-purchase items. Example:

```
% total < total.in
$13.12
%
```

Details:

Prices will be something like `$15` (fifteen dollars) or `10c` (ten cents). Prices are always integer amounts—there won't be a price like `$10.15`. A price is always preceded by a dollar sign or followed by a "`c`". Prices appear at the very end of a line—the last character of the first line of input is "`0`".

The whitespace characters are always blanks, never tabs. In the "to-purchase" list, no blanks follow the item name.

Don't worry about formatting—just print the total preceded by a dollar sign.

Assume that there is no invalid input, no duplications in the price list, and that all items named in the to-purchase list have appeared in the price list.

(Problem 4 — space for answer)

For reference: (copied from previous page)

```
% cat total.in
Icon book                       $10
Old lamp with bad switch         $2
Bowling ball                     $1
Perl book                        2c
Pencil                           6c
.
Pencil
Bowling ball
Pencil
Icon book
Old lamp with bad switch
% total < total.in
$13.12
%
```

Problem 5: (7 points)

Write a procedure `intmem(i, L)` that returns `&null` if the integer `i` is contained in the list `L` and fails otherwise. `L` may contain values of any type.

Examples:

```
][ intmem(1, [3, 1, 4, 2, 5]);
   r := &null  (null)

][ intmem(1, [3, "abc", &digits, 1, []]);
   r := &null  (null)

][ intmem(1, [3, "abc", &digits, 10, []]);
Failure

][ intmem(1, []);
Failure
```

Only the top-level elements of `L` need be considered. Do not search for `i` in lists contained in `L`:

```
][ intmem(1, [[1]]);
Failure
```

Note that a comparison such as `1 = []` produces a run-time error.

Problem 6: (10 points)

Write a program `sumints` that reads lines on standard input and prints the sum of all integers found.

Example:

```
% sumints
On February 14, 1912, Arizona became
the 48th state.
^D (control-D)
1974
%
```

A string such as `12.34` is simply interpreted as two integers: 12 and 34.

Restriction: Your solution must be based on string scanning. The only types you may use are integers, strings, and character sets. You may not any comparison operators such as ==.

Problem 7: (6 points)

According to the instructor, what is the unique aspect of Icon's expression evaluation mechanism?

Name one thing that the instructor doesn't like about Icon or has identified as a problem with the language. Here's one thing you can't mention: unexpected failure.

There are no built-in functions in Icon to do things like reverse lists, compare all elements of two lists, or do a "deep copy" of a list. What did the instructor cite as the likely reason for the absence of functions like that?

Problem 8: (8 points)

Fill in the blanks:

The instructor described the function __________ as being a "lone wolf". He said that _____ "works well with others".

The result of a successful comparison in Icon is ___________________.

We studied a total of ____ string scanning functions. Of those, two changed ___________________

and ______ of them returned a _______. ______ is one-of-a-kind.

EXTRA CREDIT SECTION (one point each)

(a) Write the result sequence of `(?"xxx" || !"xxx" || *"xxx")`

(b) If the string `s` contains your login name, what is `string(cset(s[1:4]))[1:-2]`?

(c) Which one of the following principal contributors to Icon have not been mentioned in class: Steve Wampler, Bob Alexander or Tim Korb?

(d) Cite up to three elements of Icon that are "syntactic sugar". (one point each)

(e) Given this program, `args.icn`:

```
procedure main(args)
    every write(!args)
end
```

what does `args` print when run like this: "`args < in.dat >out.dat`"?

(f) Describe a situation where `tab(n)` and `move(n)` produce the same result, for a particular subject, position, and value of `n`.