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.

```icon
procedure main()
    L := []
    while line := read() do
        if line ? find("@") then
            push(L, "<@>")
        else
            push(L, line)
        every write(!L)
end
```

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.

```icon
procedure main(a)
    x := read_file(a[1])
    y := read_file(a[2])

    if *x ~= *y then stop("Diffs")
    every i := 1 to *x do
        if x[i] ~== y[i] then
            stop("Diffs")
end
```

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.

A straightforward solution is this:

```icon
procedure main()
    t := table(31)
    every t[4|6|9|11] := 30
    t[2] := 28
    every m := 1 to 12 do
        every d := 1 to t[m] do
            every y := 1 to 99 do {
                s := m || "/" || d || "/" || y
                write(reverse(s) == s)
            }
end
```
With a little thought about the possibilities, the program can simplified:

```icon
procedure main()
    every m := 1 to 12 do
        every d := (1 to 9) | 11 | 22 do
            every y := 1 to 21 do {
                s := m || "/" || d || "/" || y
                write(reverse(s) == s)
            }
end
```

**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.

```icon
procedure main()
p := table()
while line := read() do {
    if line == "." then
        break
    reverse(line) ? {
        price := reverse(tab(upto(' ')))
        if price[-1] == "c" then
            price := price[1:-1] * .01
        else
            price := price[2:0] * 1.0
        tab(many(' '))
        item := reverse(tab(0))
p[item] := price
    }
}
total := 0
while total += p[read()]
write("\$, total)
end
```

**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.

```icon
procedure intmem(i, L)
    return integer(!L) = i & &null
end
```
Problem 6: (10 points)

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

**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 `==`.

```icon
procedure main()
    sum := 0
    while line := read() do
        line ? while tab(upto(&digits)) do
            sum += tab(many(&digits))
    write(sum)
end
```

Problem 7: (6 points)

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

*Expressions can produce zero, one, or many results.*

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.

*Some examples:*
  
  - `t := table([])` doesn't mix well with `push` and `put`
  - Semicolon insertion can lead to surprises
  - No well-defined and documented library to do things like reversing a list

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?

*The limited memory space (< 64k of compiled code) for the initial UNIX implementation and relative difficulty of writing those functions in C.*

Problem 8: (8 points)

Fill in the blanks:

The instructor described the function `move` as being a "lone wolf". He said that `tab` "works well with others".

The result of a successful comparison in Icon is the **right hand operand**.

We studied a total of **8** string scanning functions. Of those, two changed **the position** and **five** of them returned a **position**. **pos** 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 $\text{string(cset}(s[1:4]))[1:-2]$?

```icon
def s := "whm" & string(cset(s[1:4]))[1:-2];
r := "h" (string)
def s := "yourname" & string(cset(s[1:4]))[1:-2];
r := "o" (string)
```

(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)

    The unary \ and / operators

    It can be argued that the the augmented operators, such as $+:=$, are syntactic sugar.

    Two elements we didn't talk about, the repeat and case expressions, are reasonably thought of as syntactic sugar.

(e) Given this program, args.icn:

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

    what does $\text{args}$ print when run like this: "args < in.dat >out.dat"?

    *It produces no output.*

(f) Describe a situation where $\text{tab}(n)$ and $\text{move}(n)$ produce the same result, for a particular subject, position, and value of $n$.

```icon
$"\text{? tab}(2)$;  
Failure

"\text{? move}(2)$;  
Failure
```