Homework 8 — various topics — assigned Wednesday 14 April — due Sunday 2 May

Reading assignment

Read Chapters 1-3 of *Programming in Prolog*.

8.1 [Prolog] Arithmetic evaluator (15pts)

The relation `ev` is between an expression in a very restricted arithmetic language described below, and a number which that expression evaluates to, in this little language. You can assume that the first argument is instantiated.

```
?- ev(3,N).
N=3

?- ev([+,3,4], N).
N=7

?- ev([+,3,[*,8,0.25]], N).
N=5
```

Expressions in the "little language" are defined as

```
 E ::= number
     | [+, E, E]
     | [*, E, E]
     | [exp, E]
     | [sin, E]
     | [cos, E]
```

These have the obvious numeric meanings when evaluated. Write the rules for `ev`.

8.2 [Prolog] Numerals (10pts)

Define numerals as follows:

```
%% num(X) means: X is a numeral.
num(0).
num(s(X)) :- num(X).

%% sum(X, Y, Z) means: X, Y, Z are numerals
%%         such that Z is the sum of X and Y.
sum(X, 0, X) :- num(X).
sum(X, s(Y), s(Z)) :- sum(X, Y, Z).
```

What are the answers to the following queries:

1. `?- sum(s(s(0)), s(s(s(0))), Z).`
2. `?- sum(X, Y, s(s(s(0)))).`

8.3 [Prolog] Multiplication (10 pts)

Continuing the preceding exercise, write the rules for multiplication `mult(X, Y, Z)`, meaning that `Z` is the product of `X` and `Y`. Multiplication is defined by the following axioms:

- $x \cdot 0 = 0$
- $x \cdot s(y) = (x \cdot y) + x$

8.4 [Prolog] Lists (10 pts)

The following are the rules for `islist(L)`, which means that `L` is a list.

```
islist([]).
islist([_|T]) :- islist(T).
```

Write the rules for `listlength(L, X)`, which means that `X`, a numeral *as defined in the preceding exercises*, is the length of the list `L`.

8.5 [ML] Mastermind (40pts)

This exercise is on the well known puzzle game of *Mastermind*.

8.5.1 Background

As far as we know, Invicta Toys and Games Limited are the owners of the intellectual property rights for Mastermind world-wide (see `http://dspace.dial.pipex.com/town/road/gbd76/toys.htm`). Their licensee for the USA is Pressman Toy Corporation. A plastic board for Mastermind, with a code length of 4 and with 6 colors, manufactured by Pressman Toy Corporation (`http://www.pressmangames.com/`), is sold at Toys'R'Us.

There is a JavaScript program with a nice and intuitive user interface by Keith Drakard that allows one to play Mastermind at `http://www.irt.org/games/js/mind/`. Note, however, that the program is buggy. There is a problem with the option "Mark in ball position?". If checked, it is supposed to force indicators into the positions of the balls (this is nonstandard). However, the positions are sometimes indicated inconsistently, as far as we can tell. Also, even without this option, sometimes the evaluator returns a white when it should return a black; this seems to occur when "Allow same colours?" is checked. A much less flexible but apparently logically correct Java applet by Karl Hörnell can be found at `http://www.javaonthebrain.com/java/mastermind/`. If you find other versions of Mastermind on the web, let us know.

Ben Andrews `<bandrews@cs.unm.edu>` has debugged the JavaScript program. His version can be found at
`http://www.cs.unm.edu/~bandrews/mind/`.

8.5.2 Formal description of the game

In this exercise, we assume the following options settings: number of balls in play, i.e., number of colors, is C; code length is L; blanks are not allowed (a blank is equivalent to another color, so this is not a true restriction); repeated same colors are allowed in the code; the response simply consists of two numbers, the number of blacks and the number of whites; there is no limit on the number of attempts.

We define the following protocol for communication between the code maker (the server) and the code breaker (the client).

After an initial handshake (which ensures that the server and the client agree on the values of C and L), the client sends the server a code guess, and the server responds with an evaluation, and this exchange is then repeated.

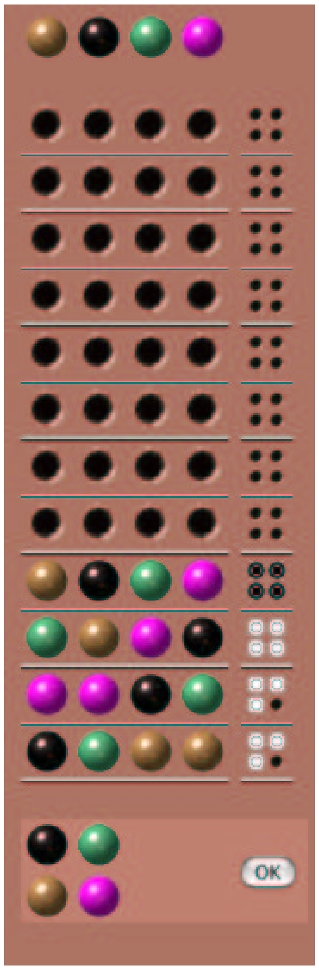
In the initial handshake, the client chooses the values of C and L, and sends them to the server. Both values are integers, and $1 \leq C \leq C_m$, and $1 \leq L \leq L_m$, where C_m is the maximum allowed number of colors and L_m is the maximum allowed code length. The values of C_m and L_m are implementation-dependent, and are known to both the client and the server, so there is no need for the client and the server to agree on them during the handshake. Having received C and L, the server randomly generates a code of length L using C colors, and sends 0 as acknowledgment. The colors are encoded as integers in the range $1,\ldots,C$. Thus, the server maintains a secret L-tuple of integers in the range $1,\ldots,C$.

The client's guess is also an L-tuple of integers in the range $1,\ldots,C$.

The server's response is a pair of integers (b,w), where b is the number of black pegs returned in the board game and w is the number of white pegs. Both are integers in the range $0,\ldots,L$. The response is computed as follows: b is the number of positions in the secret L-tuple that are equal to the corresponding position in the guess L-tuple. Removing all these positions from consideration, w is computed as the number of pairwise matchings between the rest of the secret L-tuple and the rest of the guess L-tuple, in which each element of the secret L-tuple can participate in at most one matching and eac element of the guess L-tuple can participate in at most one matching. Note that $0 \leq b+w \leq L$.

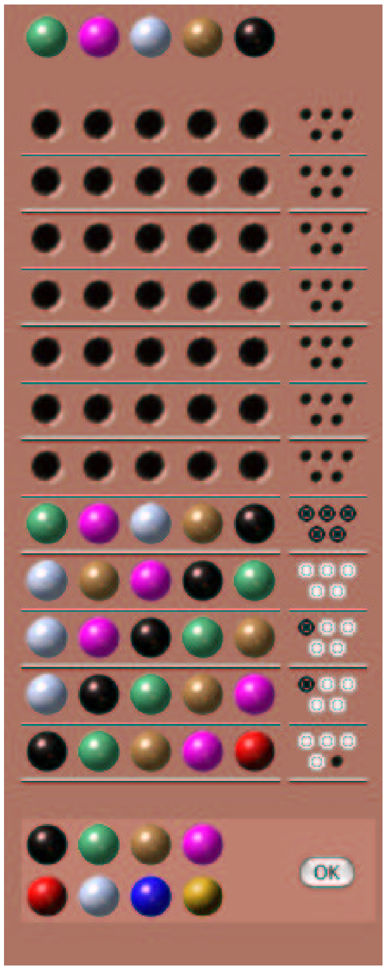
In this representation, the response $(L,0)$ means that the client's guess was correct and the game is over. The client then disconnects from the server.

Example 1. Here is a picture of a game with $C = 4$, $L = 4$, and the secret tuple 3 1 2 4, as played with Drakard's JavaScript program, and the trace of the game, as in our exercise.



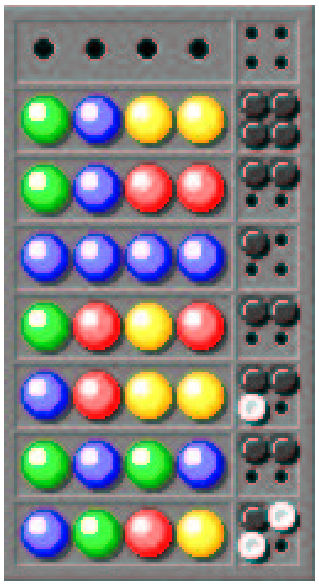
client server
4 4 $\longrightarrow$
$\longleftarrow$ 0
1 2 3 3 $\longrightarrow$
$\longleftarrow$ 0 3
4 4 1 2 $\longrightarrow$
$\longleftarrow$ 0 3
2 3 4 1 $\longrightarrow$
$\longleftarrow$ 0 4
3 1 2 4 $\longrightarrow$
$\longleftarrow$ 4 0

Example 2. Here is a picture of a game with $C = 8$, $L = 5$, and the secret tuple 2 4 6 3 1, as played with Drakard's JavaScript program, and the trace of the game, as in our exercise.



```
client      server
8 5 ⟶
⟵ 0
1 2 3 4 5 ⟶
⟵ 0 4
6 1 2 3 4 ⟶
⟵ 1 4
6 4 1 2 3 ⟶
⟵ 1 4
6 3 4 1 2 ⟶
⟵ 0 5
2 4 6 3 1 ⟶
⟵ 5 0
```

Example 3. Here is a picture of a game with $C = 4$, $L = 4$, and the secret tuple 2 1 4 4, as played with Hörnell's Java applet, and the trace of the game, as in our exercise.



client server
4 4 $\longrightarrow$
$\longleftarrow$ 0
1 2 3 4 $\longrightarrow$
$\longleftarrow$ 1 2
2 1 2 1 $\longrightarrow$
$\longleftarrow$ 2 0
1 3 4 4 $\longrightarrow$
$\longleftarrow$ 2 1
2 3 4 3 $\longrightarrow$
$\longleftarrow$ 2 0
1 1 1 1 $\longrightarrow$
$\longleftarrow$ 1 0
2 1 3 3 $\longrightarrow$
$\longleftarrow$ 2 0
2 1 4 4 $\longrightarrow$
$\longleftarrow$ 4 0

8.5.3 The task

Analysis (10pts) (a) What is the number of possible secrets given C colors and L positions? (b) Same, if repeats are disallowed? (c) How much information does the guesser gain from a (b, w) score? (d) What is the minimum expected number of guesses needed to solve the puzzle? (e) Describe a strategy that achieves that minimum, i.e., an optimal strategy.

Programming (30pts) You are given a tar file with ML implementations for the server and a driver that takes care of the protocol. Note the use of the module language and the compilation manager.

You are to implement a client.

You must not make any changes to the signatures, or to any structures or functors other than the client.

The tester module, also provided to you, will allow you to measure how well your client is doing. The tester constructs random secrets for given C and L, and it calculates how many guesses the client needed on average. Your goal is to minimize this average. (Alternatively, for small C and L, you can construct *all* possible secrets and run your client on all of them.)

The grading for this exercise will be competitive.

You will receive 5 points for any correct solution. A correct client is one that is guaranteed to solve the puzzle in a bounded number of guesses.

The TA will evaluate all submitted solutions, including his own, using the tester, and compute average numbers of guesses for each, for a range of C and L values.

Of particular interest are the values $C = 6, L = 4$ (the original Mastermind) and $C = 8, L = 5$ (Super Mastermind). The averages for these two will be added up; let's call that sum s. Let $s^\star$ be the smallest s in the set of submitted solutions. Then the number of points awarded will be $\frac{15s^\star}{s}$.

Finally, we are interested in how the solution scales to larger values of C and L. The client should be able to handle at least $C = 10, L = 10$, and up to 10 points will be awarded according to the average number of guesses for that configuration, as in the preceding paragraph.

8.5.4 Extra credit

Assumes 361/461 background.

Give an analysis of the number of guesses made (expected, lower and upper bounds) by your client. Then consider any auxiliary data structure you are using and their access times, and analyze the running time as opposed to just the number of guesses.

8.5.5 Extra credit

Assumes 241/341/481 background. Do not attempt this extra credit part unless you have completely solved the base exercise.

Instead of having a driver orchestrate the exchange of information between a client module and a server module, all inside a single ML program, here we write truly separate client and server programs.

- Write a server and a client, both in ML, using Internet sockets (see `structure INetSock`). Test with the server and the client running on the same, and on different hosts.
- Same, but in C or Java.
- Same, but so that a single server can service multiple clients simultaneously.
- Add a graphical user interface to the client.

8.6 [Prolog] Mastermind (15pts)

Write the rules for `score/3`, such that `score(Secret,Guess,Black,White)` has the obvious meaning for scoring a Mastermind guess, as described in the preceding exercise; the secret and the guess are equal-length instantiated lists of atoms, and Black and White are Prolog built-in integers that the predicate will compute. For instance, the query

```
?-score([green,purple,white,brown,black],
[white,black,green,brown,purple],Black,White).
```

should result in

```
Black=1 White=4
```

as in Example 2 of preceding exercise.

How to turn in

Make sure that you have thoroughly tested your code, and include all your test runs!

Turn in your code by running

˜clint/handin your-file

on a regular UNM CS machine. You should use whatever filename is appropriate in place of your-file.

Include the following statement with your submission, signed and dated:
I pledge my honor that in the preparation of this assignment I have complied with the University of New Mexico Board of Regents' Policy Manual, including Section 4.8, Academic Dishonesty.