HERIOT WATT UNIVERSITY

SCHOOL OF JAMIE'S SAMPLE QUESTIONS

Computer Science

F28PL

Programming Languages

Semester 1 2013/14

Well before 18th December 2013

Duration: As long as you like

ANSWER ALL QUESTIONS

Answer each question in legible writing
Respect your teachers, the poor hardworking dears
Revise well in advance
Do something nice for the Christmas break

Q3

(a) Consider the following Standard ML function:

```
fun foldl f [] empval
        = empval
  | foldl f (h::t) empval
        = f(h,foldl f t empval);
```

State `foldl`'s type. (3)

(b) State, giving full types, the output of running the following programs:

(i) `foldl (fn (x,y) => x+y) [] 0;` (2)

(ii) `foldl (fn (x,y) => x+y) [1,2,3] 4;` (2)

(c) Trace execution of the program

```
foldl (fn (x,y) => x+y) [1,2,3] 4;
```

showing the variable bindings at each stage. (6)

(d) Explain in English the similarity and the difference between the following two programs: (3)

```
fn x => if (not x) then (fn _ => false)
                   else (fn y => y);
fn x => fn y => x andalso y;
```

(e) Give one reason for, and one reason against, using ML to program a backend to an online marketplace such as Amazon or eBay.
(Further reading: http://www.paulgraham.com/avg.html.) (2)

(f) State, giving full types, the behaviour of the following program (i.e. explain what function this calculates): (2)

```
fn l =>
   foldl
      (fn (f,g) => (fn x => f(g x)))
      l
      (fn x => x);
```

(bonus Q) Batman is the better superhero model than Superman for undergraduate students. Discuss. ()

Q4

(a) Explain in English what the following Prolog program calculates: (3)

```
pfunc([M],M).
pfunc([H,K|T],M):- H=<K, pfunc([H|T],M).
pfunc([H,K|T],M):- H>K,  pfunc([K|T],M).
```

(b) "ML is a typed language and Prolog is an untyped language". Explain in English what this means. (2)

(c) Trace

```
 pfunc([3,1,2],X).
```

showing the variable instantiations and sharings at each stage. (4)

(d) Compare and contrast the treatments of variables in ML, Prolog, and C. (6)

(e) Explain what is wrong with the following Prolog program, and why:

```
add(x,y,z) :- z is x+y.
```

(2)

(f) Explain in English what the Prolog program `fun1` does:

```
fun1(X,[H|T],[H|U]) :- fun1(X,T,U).
fun1(X,[],[X]).
```

(3)

(bonus question) An attractive first year student stops you in the corridor and asks you what the difference is between Prolog and C. Impress him/her with your knowledge by giving three differences, with explanation. ()

END OF PAPER

SAMPLE ANSWERS
(Unlikely to be included in the real exam script.)

Q3(a) ('a*'b → 'b) → 'a list → 'b → 'b
Easy marks, if the student doesn't get lost in the higher-order type system.
(Some students will panic right there. Functions? Nobody told me we had to learn about functions! You can see it in the shaky writing and the tearstains on the page. Don't Panic! Go and revise functions. Now, before it's too late.)

Q3(b) 0:int and 10:int
(Some students will fail to read the question and neglect to give types. Bang! You lost two marks. How depressing.)

Q3(c)
For brevity write `adfun` for `(fn (x,y) => x+y))`.

```
foldl adfun [1,2,3] 4

     f == adfun, h == 1, t == [2,3], empval = 4
     ---> (clause 2)

adfun(1,foldl adfun [2,3] 4) = 1 + (adfun [2,3] 4)

     f == adfun, h == 2, t == [3], empval = 4
     ---> (clause 2)

1 + adfun(2,foldl adfun [3] 4) = 3 + (foldl adfun [3] 4)

     f == adfun, h == 3, t == [], empval = 4
     ---> (clause 2)

3 + adfun(3,foldl adfun [] 4)

     f == adfun, empval = 4
     ---> (clause 1)

3 + 3 + 4 =  10
```

Now that's a lot of writing --- but it's more free marks, provided you can keep a level head. Don't annoy the examiner by scrawling some crazy mess on the page. Keep it neat and make sure to look like you're in control.

Q3(d)
Both programs evaluate logical conjunction.
The second program forces evaluation of the expression bound to x and then that bound to y.
The first program forces evaluation of the expression bound to x and *only if* the value returned is `true` does it evaluate the expression bound to y.
Now we're on to slightly harder questions.

Q3(e) Any one from each of the following (or whatever else you can think of):

ML will be better for fast development and is likely to be less error-prone (and development speed and accuracy really matter if you're trying to win the races to become established that Amazon and eBay won). The programs are also likely to be more easily parallelisable.

It's harder to find ML programmers than, say, Java programmers. The resulting programs may be harder to optimise for speed, which may become an issue. Even if we do use functional programming, ML specifically may not be the best choice of functional programming language (consider Scala, Erlang, Haskell, F#).

We probably want quite a high-level language to do something like this. Raw speed is not critical in such an application: better to buy more servers.

Q3(f) Concatenate a list of functions.
Now this is what I call a real exam question: short elegant answer not easy to arrive at.

Q3(g)
Batman got where he is today through hard work, preparation, and training (and a lot of money). This is the American dream, albeit projected onto a vigilante plot.

Superman was born super. That's it. If you're not born super, you're little people. Worse: the one human who really transcended the limitations of their humanity to give Superman any real trouble---Lex Luther---was a conniving evil capitalist bastard. The suggestion being that humans that do not know their place and try to rise to greatness, will be corrupted by their genius and must be destroyed.

The character of Lois Lane exists primarily to *document* how super Superman is, so we can read about him. Batman is far too cool to care about whether we like him or not, or even know about him or not.

Of the two, the former is a model of modesty and self-empowerment (even if he did start off fithy rich, and ended up in a rubber batsuit beating people up, and breaks the law), whereas the latter is a flying power fantasy of depression-era blue-collar workers.

So the question is now this: how does Judge Dredd fit into all that?

Q4(a) pfunc(L,X) will set X to be the minimum of the list of numbers L.
Easy when you know how.

Q4(b) ML variables are associated to an assertion called a “type”, and a simple logic is used to ensure that these “types” match up *at compile time*. Prolog has none of these checks at compile time.
This is not to say that Prolog cannot raise type-like errors during runtime; for instance the “is” keyword will complain if you do not feed it an arithmetic expression to evaluate. However, this is viewed within Prolog as just another runtime error.
A tough question; many students will “know” the answer but be unable to put it into words. So let's ask: what's the essence of the student's problem here? Expressing in words what a type system is, so we can say that ML has one and Prolog does not.
And why is that difficult? Because the student needs to get their head around the difference between compile-time errors and runtime errors. So really, this question is testing: “Do you know, and can you explain, the difference between compile-time and runtime?”. If you do, you'll muddle through this question even if U right in txtspeak & cant spell.

Q4(c) *Note that, unlike with ML, variables get “dashed” as evaluation proceeds.*
I have elided the backtracking that occurs as e.g. the Prolog engine tries and fails to apply clauses 1 and 2 in the first step, and clause 1 in the second step.

```
pfunc([3,1,2],X)

     H==3, K==1, T==[2], X shares with M
     ---> (clause 3)

pfunc([1,2],M)

     H'==2, K'==1, T'==[], M shares with M'
     ---> (clause 2)

pfunc([1],1)

     M' instantiated to 1
     ---> (clause 1)

X = 1.
```

Q4(d) *Six points for this, so give the examiner six distinct things to tick. Here are the obvious things you're supposed to come up with:*
ML variables: strongly typed, allocated to expressions at most once.
Prolog variables: weakly typed, may be unbound from expressions by backtracking but not otherwise reallocated.
C variables: quite strongly typed (though pointers mess that up in practice a little), may be bound to expressions then rebound indefinitely.

Other things I can think of:
In C, you can always obtain a memory address for where the value bound to a variable is stored (i.e. get a pointer to it). Not so in ML or Prolog.

In ML and C scoping is pretty strict. In Prolog scoping is slightly more fluid.
In ML and C you cannot access the name of a variable from within the language itself. Prolog can dynamically capture the syntax of a Prolog program and examine its own syntax, including the names of variables.

Q4(e) You used lowercase letters, so these are constants. This is obviously intended to calcuate "z is the arithmetic result of adding x to y", so we should use variables, which must start with an uppercase letter.
There are two points for this question, so try to give the examiner two things to tick: e.g. any two of "constants" and "variables" and "uppercase letter". Just writing "You used lowercase letters" might not be enough, because that provides no evidence that you know why that is a bad thing here!

Q4(f) U is T concatenated with [X].
Short answer, not that easy to arrive at.

Q4(bonus question)
Please provide bonus answer here.