SCHOOL OF MATHEMATICAL AND COMPUTER SCIENCES

Department of Computer Science

F28PL

PROGRAMMING LANGUAGES

Semester 1— 201718

Duration: Two Hours

ANSWER THREE QUESTIONS

1. (a) Write ML code of the following types:

1. `int` (1)
2. `real` (1)
3. `bool` (1)
4. `int list` (1)
5. `'a list` (2)
6. `'a list -> 'a` (2)
7. `''a list -> ''a` (2)

(b) The following function $ziplist$, if given two lists l_1 and l_2 of the same length, will return a value as follows:

$$ziplist(nil, nil) = nil$$
$$ziplist(l_1, l_2) = (hd(l_1), hd(l_2)) :: ziplist(tl(l_1), tl(l_2))$$

Above, nil denotes the empty list; and $::$ denotes *list cons* (or 'push', using stack terminology); and hd denotes the head of a list; and tl denotes its tail.

1. Write an ML function `ziplist` to implement $ziplist$. For full marks, your answer must use ML pattern-matching.
2. State the type of `ziplist`. (4)

(c) Explain in English the meaning of the following two ML types, and write ML code of each type.

1. `(int -> 'a) -> 'a`
2. `int -> ('a -> 'a)`

(4)

(d) State the type of the following ML program, and explain in English what it calculates.

```
fun mystery f x =
if (f x = x) then x else (mystery f (f x));
```

(2)

2. (a) Assume a variable

```
x = ["MEGA","Encrypted", "Global","Access"]
```

State and explain in detail the behaviour and output of the following short programs (stating the output without explaining it may score no marks):

1. `x[0]`
2. `x[-4]`
3. `"".join([i[0] for i in x])` (3)

(b) 1. Precisely describe and explain the execution of the following code:

```
x = [[],[]]
x[0] is x[1]
x[0] == x[1]
x[0] = x[1]
x[0] is x[1]
x[0] == x[1]
```

(3)

2. The execution is different if we start it with `x = ((),())` instead of `x = [[],[]]`. How, and why? (1)

(c) The **map** function inputs a pair (f, l) of a function f and a list $l = [l_1, \ldots, l_n]$ and outputs the list $[f(l_1), \ldots, f(l_n)]$. Implement map . . .

1. . . . as an **iterative** function `mapi(f,l)`. (3)
2. . . . as a **recursive** function `mapr(f,l)`. (3)
3. . . . using a **list comprehension** in a function `mapl(f,l)`. (3)
4. State and explain the expected behaviour if we call `mapr(lambda x:x,range(1000))`. (1)

Your answers must be functions: so they must either open with `def` and contain a `return`, or be written using the `lambda`-construct.

(d) Explain what `f` does in a clear manner suitable (for instance) for a clear technical documentation file.

```
f = lambda l : l if len(l)<=1 else
f([x for x in l[1:] if x < l[0]]) + [l[0]] +
f([x for x in l[1:] if x >= l[0]])
```

(3)

3. Answers to these essay-style questions must be clear, specific, detailed, and also legible.

(a) Explain in detail and with justification to what extent ML, Python, and Prolog can be viewed as imperative, functional, and logic programming languages.

This question is worth 6 marks, so your answer should include (at least) six distinct, specific, easily-ticked points. (6)

(b) Explain what the Church-Rosser property is, and give two ways in which it can be a practically useful language feature. (4)

(c) Recommend, with justification, a programming language to the following people (most marks will be awarded for clear and well-informed justifications). Your suggestions may include languages not taught on this course, so long as you explain yourself clearly:

1. Me, writing a quick script to find all my .jpeg files and resize them.
2. You, implementing a mathematical function on lists from a clearly-written specification such as in Q1b of this exam paper (just writing 'ML' may score zero points; the marks are for the explanation).
3. The designer of a highly parallel, highly complex mathematical algorithm, to be run on a GPU (a graphics card).
4. Programming a rule-based system, such as the prerequisites structure of the modules on your degree.
5. Implementing a Make system for a compiler (a Make system is a system for expressing and implementing rules for compiling a program from one or more source files, to one or more target architectures).

(5)

(d) Discuss, in detail, the type systems in ML, Python, and Prolog. (5)

4. (a) 1. State what the Prolog function `mystery` does in a clear manner suitable (for instance) for a clear technical documentation file.

```
mystery([],[]).
mystery([X|L'],[X|[Y|L]]) :- mystery(L',L).
```

(4)

2. Describe, with specific and detailed reference to the Prolog execution model, the execution paths of `mystery([])`, `mystery([1,2,3])`, and `mystery([1,2,3,4])`. (6)

(b) The following function *ziplist*, if given two lists l_1 and l_2 of the same length, will return a value as follows:

$$\begin{aligned} ziplist(nil, nil) &= nil \\ ziplist(l_1, l_2) &= [hd(l_1), hd(l_2)] :: ziplist(tl(l_1), tl(l_2)) \end{aligned}$$

Above, nil denotes the empty list; and $::$ denotes *list cons* (or 'push', using stack terminology); and hd denotes the head of a list; and tl denotes its tail; and square brackets form a list, so that $ziplist$ above generates a list of two-element lists.

Implement $ziplist$ as a 3-argument Prolog predicate `ziplist/3`.

Note that the Prolog syntax for the two-element list $[1, 2]$ is `[1,2]`.

(5)

(c) Implement a 2-argument Prolog predicate `max(I,L)` such that if L is a nonempty list of numbers then I is its greatest element. (5)

END OF PAPER