HERIOT WATT UNIVERSITY

SCHOOL OF MATHEMATICAL AND COMPUTER SCIENCES

Department of Computer Science

F28PL

PROGRAMMING LANGUAGES

Semester 1— 201617

Duration: Two Hours

ANSWER THREE QUESTIONS

Answer each question in a separate script book.

1. (a) For each of the ML programs below, write its type or explain why it cannot have one:

1. `fn x => fn y => x * y`
2. `fn x => fn y => (x,y)`
3. `fn x => fn y => [x,y]`
4. `fn x => fn y => (x y)`
5. `fn x => fn y => x o y` (10)

(b) The following function $set2nat$, if given a list L of distinct nonnegative integers (that is, numbers 0, 1, 2, ...) will return a number:

- If L is the empty list then $set2nat\, L$ is zero.
- If L is non-empty and its first element is x, then $set2nat$ returns 2^x added to $set2nat$ applied to the rest of L.

(Intuitively, $set2nat\, L$ is that number such that the xth binary digit of $set2nat\, L$ is 1 precisely when x appears in L.)

Translate the specification above into correct ML code for a function `set2nat : int list -> int`.

You may assume a function `exp : int -> int` such that `exp x` corresponds to calculating 2^x. **Do not** write error-handling code for the case that the input list contains negative numbers. (4)

(c) Give a detailed and specific explanation in English of each of the ML concepts below. You may wish to illustrate your explanation with example code, if that helps to demonstrate understanding:

1. Partial application. (2)
2. Equality type. (2)

(d) For the ML program below, write its type or explain why it cannot have one. Your answer must include clear working and justification:

- `fn g => fn f => g(g o f)` (2)

2. (a) State and explain in detail the behaviour and output of the following short programs (based on a popular song):

1. `["ABC","It's easy as", "1 2 3"][1]`
2. `["ABC","It's easy as", "1 2 3"][1][2]`
3. `["ABC","It's easy as", "1 2 3"][1][2:-3]` (3)

(b) Explain in detail the difference Python makes between *mutable* and *immutable* types, and the significance of each. Include, with a clear explanation, at least one example of each. (3)

(c) 1. State and explain the behaviour and output of the following short programs:
   - Program 1. `x = y = [] ; x is y`
   - Program 2. `x = [] ; y = [] ; x is y` (3)
2. The behaviour changes if we replace `[]` with `()`. How, and why? (1)

(d) A string `l` is a **palindrome** when:

- `l` is equal to the empty string `""`, or
- `l` is equal to a single character, or
- the first element of `l` is equal to the final element of `l` and the middle part of `l` (obtained by removing the first and final elements from `l`) is a palindrome.

1. Write a **recursive** Python function `palr` that if given a string `l` returns `True` if `l` is a palindrome, and `False` if not. (3)
2. Write an **iterative** Python function `pali` that if given a string `l` returns `True` if `l` is a palindrome, and `False` if not. (3)
3. State what behaviour we might expect to see if we call `palr(" "*1000)`, and explain your answer. (1)

Note that if `l` is a string then `len(l)` returns the length of `l` and `range(n)` returns an iterator over 0 to $n-1$ inclusive.

(e) State what `f` and `g` below do in a clear manner suitable (for instance) for a clear technical documentation file.

```
f = lambda z: sum(map(lambda x:2**x,z))
g = lambda z: [i for i in range(z) if (2**i) & z]
```

(3)

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

(a) Explain the meanings of the two phrases *declarative programming* and *imperative programming*. State with detailed justification to what extent each of these two phrases applies to ML, Python, and Prolog. (6)

(b) Explain what a type system is. Discuss the role type systems play in language design, and the pros and cons of having types in a language.
For full marks, your discussion must include specific reference of ML, Python, and Prolog and demonstrate an understanding of the essential flavour of the type system in each language. (6)

(c) Each of the following four assertions has arguments for and against:

1. ML is an easy language!
2. ML is a hard language!
3. Python is an easy language!
4. Python is a hard language!

By comparing and contrasting ML and Python, give a detailed discussion of and justification for each assertion.

This question is worth 8 marks, so your answer should include at least eight distinct, specific, convincing points. (8)

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

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

(3)

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

(b) 1. Discuss the difference between the *static* and the *dynamic* databases in Prolog. Your answer should include a detailed and specific explanation of the keywords `assert` and `retract`, including any restrictions on their use. (3)

2. To what extent is Prolog a declarative language? Explain your answer in specific detail. (3)

(c) The *factorial* function $m!$ is defined on nonnegative numbers (0, 1, 2, ...) by

$$0! = 1 \qquad (m+1)! = (m+1) * (m!)$$

Implement this as a 2-argument Prolog predicate `fact(X,Y)` such that

- $fact(m, n)$ returns `Yes` if $n = m!$, and
- $fact(m, n)$ returns `No` otherwise.

(4)

(d) Implement a 3-argument Prolog predicate `range(X,Y,L)` such that L is the list of elements between X and Y inclusive.
For instance, the query `range(4,6,L)` should return `L = [4,5,6].`

(4)

END OF PAPER