UNIVERSITY OF LONDON

GOLDSMITHS COLLEGE

B. Sc. Examination 2002

COMPUTING AND INFORMATION SYSTEMS

IS52006A (CIS212)
Programming: Advanced Topics and Techniques

Duration: 3 hours

Date and time:

Answer SIX questions.

Full marks will be awarded for complete answers to SIX questions.

You must answer THREE questions from section A and THREE questions from section B. You must answer at least ONE question on Prolog in Section B.

There are 150 marks on this paper.

Electronic calculator may be used. The make and model should be specified on the script and the calculator must not be programmed prior to the examination.

Section B

Question 6

(a) Express the following lists in terms of :: and nil in Standard ML.

(i) [1, 2]
(ii) [[1], 2]
(iii) [[[1]]] [6]

(b) Define a Standard ML function *mean* that takes two integers and returns their average. Thus, for example, mean(1, 5) should return 3 and mean(3, 6) should return 5. [2]

(c) Define a Standard ML function *empty* that takes a list and returns true if and only if the list is empty. [2]

(d) Define a Standard ML function *triple* that takes a list of integers and triples each of the integers in that list. For example, triple([1, 2, 3]) should return [2, 4, 6]. [3]

(e) The function *f* is defined by the following: [5]

```
fun f(nil) = 0 |
    f(h::t) = (h mod 2) + f(t);
```

Give the step-by-step evaluation of f([1, 2, 3]).

(f) Define a Standard ML function *powerL* that takes a list *x* of integers and a number *n* and returns the list containing the elements of *x* raised to the power of *n*. Thus, for example, powerL([1, 2, 3], 2) would evaluate to [1, 4, 9]. **Hint:** you may first wish to define a function *power* that takes two integers *u* and *v* and returns *u* raised to the power of *v*. [7]

Question 7

(a) Give the step-by-step evaluation of the following expressions in Standard ML:

(i) 2*5-3
(ii) 2+4*5=7*3
(iii) if 3*5+1=15 then 1+2*3 else 2*3+4 [6]

(b) Define a Standard ML function *head* that takes a list and returns the head of that list. For example, head([3, 2, 1]) should return 3. [2]

(c) Define a Standard ML function *less_than* that takes two integers and returns true if and only if the first integer is less than the second. [2]

(d) Define a Standard ML function *increase* that takes an integer list and increase each value within it by 5. For example, increase([1, 2, 3]) should return [6, 7, 8]. [3]

(e) (i) Define a Standard ML function *length* that takes a list and returns its length. For example, length([4, 5, 6]) should return 3. [2]

(ii) Define a Standard ML function *longer_than* that takes two lists and returns true if and only if the first list is longer than the second. For example, longer_than([1, 7, 3], [5, 2]) should return true whereas longer_than([6], [3, 5]) should return false. [3]

(f) (i) Define a Standard ML function *found* that takes an integer and a list and determines whether the integer is found in the list. For example, found(2, [1, 2, 3]) should return true while found(4, [1, 2, 3]) should return false. [3]

(ii) Having defined *found*, give the step-by-step evaluation of the expression:

found(3, [2, 3, 1, 4]) [4]

Question 8

(a) What does it mean to say that Standard ML is strongly typed? [3]

(b) What is meant by the term 'constructors' in Standard ML? Give the constructors for the *bool* type, and for the *list* type. [5]

(c) (i) Explain the rules of *empty* and *add* in the following definition of a datatype, illustrating your answer by showing how such a structure containing the numbers 1, 2, and 3 could be represented:

datatype set = empty | add of int * set; [4]

(ii) Define a Standard ML function *end* that takes an integer *x* and a set *y* and adds *x* to the end of *y*. [4]

(d) Define a Standard ML function *last* that takes a list of integers and returns the last integer in the list. For example, last([1, 2, 3]) should return 3. [3]

(e) Write brief notes on Polymorphism and Overloading, explaining the differences between them using the examples *append* and <. [6]

Question 9

(a) What does it mean for two Prolog terms to match? In your explanation **give** the rules for matching in Prolog. [4]

(b) Determine the results of the following queries in Prolog. Explain your answers.

?- admires(john, X) = hates(Y, mary).
?- likes([pat, sue], [tom, jim, bob]) = likes(X, [Y|Z]).

[5]

(c) Define left-recursion and explain the problem it can cause. **Illustrate** your answer with an example. [5]

(d) Define a Prolog predicate *add* that takes three arguments X, Y and Z such that Z is the result of adding X to Y. For example, add(3, 5, 8) should return *Yes* whereas add(3, 4, 8) should return *No*. [2]

(e) Define a Prolog predicate *only_one* that takes a list and returns *Yes* if and only if the list contains exactly one element. For example, only_one([a]) should return *Yes* whereas only_one([a, b, c]) should return *No.* [2]

(f) Suppose the following have been given:

1) male(john).
2) male(steve).
3) female(mary).
4) married(john).
5) married(mary).
6) unmarried(steve).
7) bachelor(X):- male(X), unmarried(X).

Give the step-by-step evaluation of the following queries in terms of unification and goal replacement:

?- bachelor(steve). [3]
?- bachelor(john). [4]

Question 10

(a) For each of the following lists, represent the list using the dot functor:

(i) [a]
(ii) [[a]]
(iii) [[a], b] [5]

(a) (i) Explain the role of *backtracking* in Prolog. [3]
(ii) Explain the effect of the **cut** on backtracking. [2]

(b) Using predicates *is_city*, *is_beautiful*, and *is_beautiful_city* write Prolog rules and facts that state:

- London is a city.
- London is beautiful.
- If something is a city and is beautiful then it is a beautiful city.

Illustrate your answer by explaining the execution of a query that asks 'Is London a beautiful city?'. [6]

(c) Define a Prolog predicate *sum2* that takes a list L of integers and an integer N and returns *Yes* if and only if N is the result of adding up all the **positive** integers in L. For example, sum2([1, -2, 3], 4) should return *Yes* while sum([1, -2, 3], 2) should return *No.* [4]

(d) Define a Prolog predicate *remove_last* that takes two lists and returns Yes if and only if the second list is the result of removing the last element from the first list. For example, remove_last([a, b, c], [a, b]) should return *Yes* whereas remove_last([a, b, c], [a, c]) should return *No.* [3]

(e) Define a Prolog predicate *second* that takes a list L and an item X and returns *Yes* if and only X is the second item in L. For example, second([a, b, c], b) should return *Yes* whereas second([a, b, c], c) should return *No.* [2]