UNIVERSITY OF LONDON

GOLDSMITHS COLLEGE

B. Sc. Examination 2003

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], 2] [6]

(b) Define a Standard ML function *even* that takes an integer and returns *true* if and only the integer is an even number. [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 multiplies each of the integers by three. For example, triple([1, 2, 3]) should return [3, 6,9]. [3]

(e) The function *f* is defined as follows:

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

Give a step-by-step evaluation of f([1, 2, 3]). [5]

(f) Define a Standard ML function *factorialL* that takes a list *x* of integers and returns the list containing the factorials of all the elements in *x*. For example, factorialL([3, 4, 5]) would evaluate to [6, 24, 120]. **Hint:** you may first wish to define a function *factorial* that takes an integer and returns its factorial. [7]

Question 7

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

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

(c) Define a Standard ML function *decrease* that takes an integer list and decreases each of the integers by 2. For example, decrease([4, 8, 5]) should return [2, 6, 3]. [3]

(d) (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 *twice_as_long* that takes two lists and returns *true* if and only if the first list is twice as long as the second. For example, twice_as_long([1, 7, 3, 8], [5, 2]) should return *true* whereas twice_as_long([6, 9], [5, 2]) should return *false*. [3]

(e) (i) Define a Standard ML function *product* that takes an integer list and returns the product of all the integers. For example, product([1, 2, 3, 4]) should return 24, which is the result of 1*2*3*4. [3]

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

product([4, 5, 6]) [4]

(f) Suppose that we have some records about certain people, for example, one record is: {name="bob", age=30, profession="manager", weight=150.57}.

(i) What is the type of this record? [2]
(ii) Define a function *older* that takes two persons' records and return *true* if and only if the first person is older than the second person. [4]

Question 8

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

(b) (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, 3, and 4 could be represented:

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

(ii) Define a Standard ML function *front* that takes an integer *x* and a set *y* and adds *x* to the front of *y*. [2]

(c) 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]

(d) Define a Standard ML function *squareL* that takes a list of integers and squares all the integers. For example, squareL([1, 2, 3]) should return [1, 4, 9]. [3]

(e) Define a Standard ML function *sumEven* that takes a list of integers and returns the sum of the even integers. For example, sumEven([1, 2, 4, 7]) should return 6. [4]

(f) Write brief notes on Polymorphism and Overloading, explaining the differences between them using *append* and < as examples. [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 *only_two* that takes a list and returns *Yes* if and only if the list contains exactly two elements. For example, only_two([a, b]) should return *Yes* whereas only_two([a, b, c]) should return *No.* [2]

(e) 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 a step-by-step evaluation of the following queries in terms of unification and goal replacement in the goal stack:

?- married(john)). [3]

?- bachelor(X). [6]

Question 10

(a) Explain the meaning of *facts*, *rules* and *queries* in Prolog, giving suitable examples [6]

(b) Without using the built-in operator *not*, define a Prolog predicate *different* that takes two items and returns *Yes* if and only if the two items are different. For example, different(a, b) should return *Yes* whereas different(a, a) should return *No.* [3]

(c) Define a Prolog predicate *sum* that takes a list *L* of integers and an integer *N* and returns *Yes* if and only if the *N* is the sum of all the integers in *L*. For example, sum([1, 2, 3], 6) should return *Yes* while sum([1, 2, 3], 10) should return *No.* [3]

(d) Define a Prolog predicate *sum2* that adds up all the **odd** integers in a list. For example, sum2([2, 3, 4, 1], 4) should return *Yes* while sum2([2, 3, 4, 1], 10) should return *No.* [4]

(e) Explain the behaviour of the functor *not* in Prolog, and discuss the difference between *Yes/No* and *true/false* in Prolog. Illustrate your answer by considering the query `?-single(clinton)`, given the following facts and rule:

```
single(bob).
married(ivy).
single(X):- not (married(X)).
```

[9]