UNIVERSITY OF LONDON

GOLDSMITHS COLLEGE

B. Sc. Examination 2010

Computer Science

IS53020A (CIS335) Logic Programming

Duration: 2 hours 15 minutes

Date and time:

There are five questions in this paper. You should answer no more than three questions. Full marks will be awarded for complete answers to a total of three questions. Each question carries 25 marks. The marks for each part of a question are indicated at the end of the part in [.] *brackets.*

There are 75 marks available on this paper.

No calculators should be used.

THIS PAPER MUST NOT BE REMOVED FROM THE EXAMINATION ROOM

Question 1 Prolog syntax, unification and datatypes

(a) Write down the most specific syntactic category, in Prolog, of the underlined part of each of the following items. For example, the most specific category of the underlined part of `p(X)` is "variable".

i. `q(2) :- q(a,2).` [1]

ii. `q(a(1,X)).` [1]

iii. `a = 1.` [1]

iv. `p(X) :- \+ q(X,Y), r(Y).` [1]

v. `p(X) :- \+ q(X,Y), r(Y).` [1]

(b) What are the effects of executing the following queries in SWI Prolog, in terms of success and failure and, where appropriate, the instantiation of variables?

i. `a(X) = a(Y).` [1]

ii. `a(X,Y) = a(1,Z), f(g(Z),2)=f(g(X),Z).` [2]

iii. `p(g(X),g(g(X))) = R(Z,Y), Y = Z.` [4]

(c) Explain the following syntactic types in Prolog in terms of atoms, terms and numbers and truth values:

i. `term` [2]

ii. `functor` [2]

(d) Explain in detail the difference between the `=/2` predicate and the `is/2` predicate, giving at least one example of a pair of arguments on which they differ, and noting any conditions under which they are not executable. [5]

(e) Write down Prolog data structures which will unify with the following data and nothing else, making all of the unified values available to further unification.

i. A list containing at least 4 items, the first being the same as the third. [2]

ii. A term whose functor is `p` and which has three arguments. [2]

Question 2 Negation and Metaprogramming

(a) Given the notions of success and failure, corresponding with truth and falsehood in logic, describe the logical meaning of the Negation as Failure (NAF) operation in Prolog. [3]

(b) What is *floundering*? [3]

(c) Give a detailed example of how floundering can happen. [10]

(d) Write down one or more further clauses which would enable the `solve/1` predicate, discussed in the course, to interpret the negation operator, `\+` , as negation as failure. You may use a cut in your solution. [9]

Question 3 Prolog predicate definition

(a) Given that `ord/1` is a predicate which succeeds when its argument is a list of numbers sorted into increasing numerical order, and that `permutation/2` is a predicate that succeeds when its second argument is any permutation of its first, what mathematical operation does the predicate `m/2`, below, fulfil? Explain your reasoning, focusing especially on any special features of Prolog that are important here. [5]

```
m( A, B ):-permutation( A, B ),
           ord( B ).
```

(b) Write down a definition of `ord/1`. It will have three clauses, of which the first is given below; you may wish to use the built-in arithmetic operator, `=</2`; your predicate will be recursive. You need to think in terms of comparing each element of the list with the next. [20]

```
ord( [] ).
```

Question 4 Prolog lists and list processing and set predicates

(a) What is the primary difference between a list and other structured datatypes from the point of view of storing data items? [2]

(b) The syntax commonly used for Prolog lists, such as `[X|Y]` is *syntactic sugar* for a less readable internal representation. What are the two constructors from which the datatype is built? [6]

(c) The predicate `member/2` can be used to test whether a term is an element of a list. It is defined thus:

```
member( H, [H|_] ).
member( X, [_|T] ) :-member( X, T ).
```

i. Using `member/2`, write a predicate, `memberboth/3`, which succeeds if its first argument is a member of both its second and third arguments, assuming they are lists. [2]

ii. Write a program which will use `memberboth/3` with a Prolog set predicate to compute the intersection (*i.e.,* the list of common members) of two lists. Call your program `intersection/3`. [5]

iii. Write another version of `memberboth/3` which does not call any other predicates. Your predicate will need 4 clauses, 3 of which will be recursive. To see how to write the new predicate, consider all the different cases generated by the two clauses of `member/2` in your answer to (i) above. [10]

Question 5 Prolog syntax, metaprogramming and cut

(a) Write down the syntactic categories of the underlined parts of the following items. For example, in `p(X)`, `X` is a "variable". Parts (ii) and (iii) have two possible answers: give both.

i. `p(C,D) :- q(C,E), w(E,D).` [1]

ii. `X is A + 1` [2]

iii. `X is A + 1` [2]

(b) Explain the function of the Prolog *cut* operator, `!`, in terms of both proof tree (you may presuppose the concept of backtracking) and variable instantiation. [9]

(c) What is the difference between "red" and "green" cuts? What useful purpose do "green" cuts serve? [4]

(d) Below is an incomplete version of the metainterpreter `solve/1`, which implements the standard Prolog execution rule. Clause 2, which is missing, deals with the case of a goal which is a conjunction of literals. Write down the missing clause. [7]

```
solve( true ).                          % Cl. 1
% Missing clause 2, line 1              % Cl. 2
% Missing clause 2, line 2
solve( Goal ) :-clause( Goal, Next ),% Cl. 3
                solve( Next ).
```