MSc Computing Science 2007-2008
Prolog Laboratory Test

Imperial College London

Thursday 3h January 14:45 -17:00

- You must complete and submit a working program by **17:00**
- Log into the Lexis exam system using your DoC login as both your login and as your password (**do not use your usual password)**.
- You are required to extend the programs in two Prolog files **sales.pl** and **rules.pl** according to the specifications overleaf.
- You will find these files in your Lexis home directory (**/exam**). If you are missing these files please alert one of the invigilators. Test your work using the suggested queries.
- **Save your work regularly.**
- The system will log you out automatically once the exam has finished. **It is therefore important that you save your work and quit your editor when you are told to stop writing.** No further action needs to be taken to submit your files – the final state of your Lexis home directory (**/exam**) will be your submission.
- No communication with any other student or with any other computer is permitted.
- You will have reading time of **15 minutes** before you are allowed to start working. You will have two hours from the time you begin working.
- You are not allowed to leave the lab during the first 30 minutes or the last 30 minutes.
- **This question paper consists of 5 pages, including this page.**

1. Given file for this question is sales.pl. Submit your extension of this file as salesE.pl

The given file contains facts for the relations:

sellsFor(S,I,P)	Supplier S sells item I for price P
inStock(S,I)	Supplier S has item I in stock
locatedIn(S,C)	Supplier S is located in city C
typeOfItem(I,T)	Item I is of type T
equivalentTo(I1,I2)	Items I1 and I2 have equivalent functionality and identifier I1 is lexically before identifier I2

and the definition of **forall/2**.

Using ***forall***, ***setof***, Prolog's negation operator **\+**, where appropriate, together with any arithmetic primitives you may need, add definitions for the following relations to the file:

i) sellsOneForLessThan(T,MP,S,I,P)

Supplier **S** sells item **I** of type **T** at a price **I** which is less than **MP**. (Assume MP will always be given in any query to this relation.)

Test your definition with the query:

sellsOneForLessThan(electricKettle,30,S,I,P)

You should get two answers, both with **S='Peter Jones'**

ii) **equivalent(I1,I2)**

I1 and I2 have equivalent functionality no matter what the alphabetic order of identifiers I1, I2.

Test you definition with the query:

equivalent(I1,I2)

You should get 4 answers.

iii) **sellsEquivalentItemIn(I,C,EI, S)**

S is a supplier located in city **C** and **S** either has item **I** in stock and **EI=I**, or **S** has an equivalent item **EI** in stock for no more than than its price for item **I**. You might find it easier to use two rules.
(**I** will be given in queries to this relation.)

Test your answer with the query:

sellsEquivalentItemIn(swan123,London,I,S)

You should get two answers.

iv) **neverUnderSold(S,C)**

There is no other supplier that in city **C** sells any item that **S** sells for a price less than **S**.

Test you definition with a query: **neverUnderSold(S,C)**

You should get one answer: **C=london, S='Peter Jones'**

v) **listOfSuppliersFor(I,C,L)**

L is a list of pairs **(P,S)** where **S** is a supplier located in city **C** that supplies item **I** for the price **P** and has **I** is in stock. **L** is ordered by increasing price – lowest price comes first. Test your answer with the query:

?- **listOfSuppliersFor(electrolux09,C,L).**

You should get the answer:
C=london
L=[(70,'Peter Jones'),(80,'Harrods')]