MSc Assessed Prolog Lab Exercise 2
Issued: 30 November 2011
Due: 16 December 2011

Do all the questions and submit via Cate one file called 'answers2.pl' containing all the programs. The submitted file should be self-contained: it should be possible to *consult* your file successfully using the lab-installed version of Sicstus Prolog and to test your programs without needing to consult additional Sicstus Prolog libraries.
See the end of this specification for how to include *directives* in your program for loading libraries.

1. Write a Prolog program that can decode a message using the following code breaking key:

Replace the word 'bear' by the word 'double'.
Replace the word 'cub' by the word 'agent'.

Your program should define a predicate
decode(Message, Decoded_Message)
and any other predicates you need for this, such that Decoded_Message is the decoded version of Message according to the above key. For example, if
Message is [bill, is, a, bear, cub, and, bill, went, shopping, with, jim] then
Decoded_Message is [bill, is, a, double, agent, and, bill, went, shopping, with, jim].
Note: In this exercise, *words* are constants. The sentences *Message* and *Decoded_Message* are therefore lists of constants.

2. Using the program in 1 or otherwise, write a Prolog program for agents/3, such that
agents(Message, Decoded_Message, ListofAgents)
produces not only the decoded version of Message but also the list of names the Message accuses of being double agents. The list should have no duplicates and should be sorted. Someone is accused of being a double agent if they are said to be a bear cub. The name of an accused individual is always one word and never clashes with the code-words (i.e. a name cannot be 'bear', 'cub', 'double' or 'agent'). Your program should also work if Message contains no accusations, in which case*, ListofAgents* should be empty.

3. Write a Prolog program for a predicate *count_word(W,L,C)* such that C is the number of times the word W occurs in list L. You can assume that in any call to count_word/3 the first two arguments are fully instantiated.

4. Using your programs in 2 and 3, or otherwise, write a Prolog program for the predicate *count_ag_names(Message, Ag_name_counts)* such that Message is a coded message and Ag_name_counts is a list of elements of the form (Name, Count) where Name is a name accused of being a double agent in Message, and Count is the number of times that name occurs in Message. For example if

Message is [bill, is, a, bear, cub, and, bill, went, shopping, with, jim, and, mary, is, a, bear, cub] then Ag_name_counts is [(bill, 2),(mary,1)]. The list Ag_name_counts should be ordered on Name. Again, your program should return an empty list if Message contains no accusations.

5. Using any of the above programs or otherwise, write a Prolog program for accusation_counts(M, AC) such that given a coded message M, AC is a list of elements of the form (Name, Count) where Name is a name accused of being a double agent, and Count is the number of times that name is accused of being a double agent in M. For example if
M is [jack, is, a, bear, cub, and, bill, is, a, bear, cub, and, bill, went, shopping, with, jim, and, mary, is, a, bear, cub, and, jack, is, a, bear, cub] then
AC is [(jack, 2), (bill, 1), (mary,1)].
The list AC should be in descending order of Count, and your program should return an empty list if M contains no accusations.

Questions 1-5 have maximum marks of 20%, 15%, 15%, 20%, 30%, respectively.

How to enter directives in your program so that libraries are loaded when your program file is consulted:

If your program requires the lists library add the following entry (called a Prolog *directive*) at the top of your file:

:- use_module(library (lists)).

This causes Sicstus to automatically load the lists library upon consulting your file.

Another directive you may find useful in the above exercises is

:- set_prolog_flag(toplevel_print_options, [max_depth(100)]).

If lists are of a certain length, Sicstus, by default, uses ellipses (i.e. abbreviates them) rather than displaying the entire list. The above is a Sicstus meta-directive for overriding this: it increases the depth (complexity) of terms before which Sicstus will use ellipses when displaying terms on the screen (i.e. this directive will display more detailed/lengthy terms without abbreviation).

You should not require any other libraries/directives.