Logic and Verification  
– Sheet 7: PROLOG –

Remarks:
- For this exercise, you will need SWI-Prolog¹.
- For exercises 1 and 2, you need to send your solutions as source files to your respective tutor.
- Make sure your source files do properly compile and run!
- Document and explain your code!

Exercise 1 (8 Points, Recursive rules)

(a) Write a PROLOG program (i.e. predicate) \( \text{sigma}/2 \) which calculates \( \sum_{i=1}^{n} i \), given an integer \( n > 0 \).
E.g. \(?- \text{sigma}(4, X) \). should yield \( X = 10 \).

(b) Change your predicate so that all intermediate and final results are stored in the database and reused whenever possible (e.g. in follow-up queries). Also, each result should only be stored once.
E.g. \(?- \text{sigma}(4, X) \). may store \( \text{res}(4, 10) \)

Exercise 2 (8 Points, Powerset)

(a) Write a Prolog program \( \text{subset}/2 \) that is satisfied when the first argument is a subset of the second argument (that is, when every element of the first argument is a member of the second argument). For example:

\[
?- \text{subset}([a,b],[a,b,c])
\]
true.

\[
?- \text{subset}([b,d],[a,b,c])
\]
false.

\[
?- \text{subset}([], [a,b,c])
\]
true.

Your program should be capable of generating all subsets of an input set by backtracking. For example, if it gets the input \(?- \text{subset}(X, [a, b, c]) \) it should successively generate all eight subsets of \([a, b, c] \).

(b) Using the subset predicate you have just written, and \( \text{findall}/3 \), write a predicate \( \text{powerset}/2 \) that takes a set as its first argument, and returns the powerset of this set as the second argument.

¹http://www.swi-prolog.org/
Exercise 3 (6 Points, Cut)

List and explain for each of the following programs the results for the goal \( ? - f(X) \).
(including any backtracking requested).

(a) \( f(X) :- X = r. \)
    \( f(X) :- !, X = p. \)
    \( f(X) :- !, X = q. \)

(b) \( f(X) :- !, g(X). \)
    \( g(X) :- X = r. \)
    \( g(X) :- X = q, !. \)

(c) \( f(X) :- g(X), !, h(X). \)
    \( g(p). \)
    \( g(q). \)
    \( h(p). \)
    \( h(q). \)

(d) \( f(X) :- !, g(X). \)
    \( g(X) :- !, h(X). \)
    \( g(X) :- X = p. \)
    \( h(X) :- !, !, X = r. \)

(e) \( f(p) :- !. \)
    \( f(q) :- !. \)
    \( f(r). \)

(f) \( f(X) :- g(X). \)
    \( f(X) :- !, X = q, !. \)
    \( g(X) :- !, X = r. \)
    \( g(X) :- !, X = s. \)

Exercise 4 (6 Points, SLD and Cut)

Consider the two programs

1. \( \text{min}(X,Y,Y) :- X \geq Y, !. \)
2. \( \text{min}(X,Y,X) :- X < Y. \)

and

1. \( \text{min}(X,Y,Z) :- X \geq Y, !, Y = Z. \)
2. \( \text{min}(X,Z,X). \)

both of which determine the minimum of two numbers.

(a) Draw the SLD trees for both programs for the goal \( ? - \text{min}(6,2,G) \). Ignore the cut! (i.e. continue drawing the tree even if the cut would remove a path) Mark in each tree all solutions that are found, all failures and all unifications.

(b) Mark in each tree where paths are removed by the cut.

(c) What is the difference between both programs? Decide for each cut, whether it is red or green and explain why.