# CO884, Assessment 2, Prolog

Marks: 20 marks per question.

1. Define a predicate **`same_content/2`** whose two arguments are two lists. It should succeed if and only if all elements of the first list occur in the second, and vice versa. Note that it is allowed if these list elements occur in a different order, or a different number of times, so `?-same_content([1,2,4,2],[4,2,1])` should succeed.
2. Define a predicate **`totatives/2`** whose first argument is a natural number (a positive integer) N, and the second argument is the result, which should be the ordered list of all number between 1 and N (inclusive) which are "relative prime" to N, meaning the greatest common divisor (gcd) of N and that number is 1. For example, `?-totatives(9,X)` should succeed with X=[1,2,4,5,7,8].

   Note that `gcd/2` is a function understood by the built-in evaluation routines, e.g. `?-X is gcd(9,6)` succeeds with X=3. Note also that you will probably need an auxiliary predicate that iterates from 1 to N to find the numbers that need to go into the result list.

3. Define a predicate **`aNatural/1`** which should succeed if its argument is a natural number. If the argument is a variable then the predicate should generate all natural numbers on backtracking. This means, **`?-aNatural(X)`** should succeed in order with X=0, X=1, X=2, etc.

   If the argument is a natural number to begin with (e.g. 1000) then we want the queries to succeed straight away (rather than to slowly count up to 1000 and succeed then). To achieve this, use meta-predicates such as `var/1`, `nonvar/1`, `integer/1`, which enquire whether their argument is or is not a variable or integer, without instantiating anything.

4. Define a predicate **`anInteger/1`** which does the corresponding thing, but with all integers, including negative ones. For this you need to consider how to get from one integer to another in such a way that it will eventually find them all. Thus, also write a predicate **`nextInteger/2`** whose first argument is an integer (the second is the result parameter). Which integer you consider as "the next" integer is up to you.
5. Write a predicate **`eval/2`** whose first argument is an arithmetic expression on integers which may or may not contain variables. The second argument is the result parameter. If the expression it encounters contains a variable the predicate should "guess" its value, using `anInteger/1` from the previous question (or: `aNatural/1` if you did not get this to work). For example, `?-eval(X*3,6)` should succeed with X=2.

   The supported operators for the arithmetic should include '+'/2, '-'/2, '*'/2, div/2 and mod/2.