Exercises for the Course

Logic Programming Engineering

(Author Dr. W. Nauber)
Dr Paola Bruscoli, Peter Steinke, Amin Timany

Practical 4

Lists, Recursion III

Exercise 4.1

Define a predicate $sum\_sublist(List, Sum, SList)$ where $SList$ is a sublist of list $List$ whose elements have the sum Sum.

Example: $? - sum\_sublist([2,1,3,4],5,SL)$. gives $SL = [2,3]$ and $SL = [1,4]$.

Exercise 4.2

Test with a predicate $prime(Number)$ whether a given natural number Number is prime. If the examined number is not prime output the first factor that is found.

Hints: Use facts for the numbers 2 and 3.
Define an auxiliary predicate $has\_factor(N, F)$ which is true if the number N has the factor F.
Use only odd numbers for F as possible factors of N.
The search should terminate as soon as $F * F > N$ for a candidate F.
For the output use the system predicates $write$ and nl.
If you want that a predicate fails use the systempredicate $fail$.

Examples: $? - prime(23)$. has answer Yes.
$? - prime(143)$. has answer No and the output: 11 is a factor of 143.

Exercise 4.3

Write a program which converts a decimal number into a string of Roman numerals with the same value.

Roman numerals:

- are letters used by the Romans for representation of cardinal numbers:
1 is represented by I, 5 by V, 10 by X, 50 by L, 100 by C, 500 by D, and 1000 by M. Other numbers are represented by the shortest sequence of these letters with the required total value: their values are added except when a letter of lower denomination precedes one of higher in which case it is subtracted from it; for example,
IV $\widehat{=}$ 4 = 5 - 1, CD $\widehat{=}$ 400 = 500 - 100, but VI $\widehat{=}$ 6 = 5 + 1, and DC $\widehat{=}$ 600 = 500 + 100 etc.
A value of a letter is substracted at most once: 8 $\widehat{=}$ VIII, but not 8 $\widehat{=}$ IIX.

Hints for realization in Prolog:

Define a predicate $roman(N, R)$ which for a given decimal number N converts its value into a string R in roman numeral system.
For converting, define an auxiliary predicate $numeral(N, NL, RL, R)$ which converts N into the string R using a conversion table which is encoded as lists NL and RL.

Begin your program with:

```
roman(N, R):-
 numeral(N,[1000, 900,500, 400,100,  90, 50,  40, 10,   9,  5,   4,  1],
           [  'M','CM','D','CD','C','XC','L','XL','X','IX','V','IV','I'],R).
```

1) Add a fact for $N = 0$ which generates the empty string: $numeral(0, ...)$.

2) Add a rule for *numeral* which recursively reduces the conversion table to $[N1|NL2], [R1|RL2]$ with $N >= N1$.

E.g. $N = 25$ leads to $N1 = 10$, $R1 =$ 'X' and the query: $numeral(25, [10, 9, 5, 4, 1], ['X','IX','V','IV','I'], R)$.

3) Add a rule for *numeral* which repeats the conversion with the remainder $N2 = N - N1$, yielding string $R2$; concatenate strings $R1$ and $R2$ to solution R using build-in predicate *atom_concat*.

Example: $? - roman(1999, R)$. should yield $R =$ 'MCMXCIX'
because 1999 = 1000 + 900 + 90 + 9 and 'MCMXCIX' = 'M' + 'CM' + 'XC' + 'IX'.