CS_125 Logic Programming – Exercises 2 (not assessed)

Question 1. Define a predicate `pythagoras/3` such that if A, B are numbers, then `pythagoras`(A, B, X) will compute the number $X = A^2 + B^2$.

Question 2. Define a predicate `nat_init/2` creating for any number `N` the list `[0,...,N]`

Question 3. Define a predicate `sum_list/2` computing for any list of numbers the sum of its members.

Question 4. Define a program for computing the *Fibonacci numbers*

$$1, 1, 2, 3, 5, 8, 13, \ldots$$

That is,

$$f_1 = f_2 = 1,$$
$$f_n = f_{n-1} + f_{n-2} \quad (n > 2)$$

Question 5. The program of question 4 will probably quite slow (try it with $N = 20$). Write a faster program using the following idea: Write an auxiliary program `fib_pair/3` that computes for given number n the Fibonacci numbers f_n and f_{n+1}. Think how you can get the n-th pair form the $n-1$-st pair, for $n > 1$.

Question 6. Define a predicate `write_sums_of_sublists/1` that writes for a given list L of numbers all its sublists and their sums on the screen.

Question 7. Implement *Quicksort*. The idea is as follows: Given a nonempty list `[X|L]` of numbers, split the tail, `L`, into two parts, `Low` and `High`, such that `Low` contains the members of `L` that are less than `X` and `High` those that are greater than or equal to `X`. Recursively sort `Low` into `Low_sorted` and `High` into `High_sorted`. The sorted version of `[X|L]` is then given by the concatenation of `Low_sorted` and `[X|High_sorted]`.