Logical reasoning and programming, lab session V

(October 30, 2017)

V.1 Check how to debug and trace programs, see Sections 4.39 and 2.9 in the reference manual of SWI-Prolog. In particular, try `trace/0` and `nodebug/0`, `spy/1` and `nospy/1`. Also `guitracer/0` can be a useful tool. By the way, if you need help try `help/1` and `apropos/1`.

V.2 Try the following queries:

```
X is 4+3-2       X = 4+3-2       X =:= 4+3-2
X-2 is 4+3-2     X-2 = 4+3-2     7-2 =:= 4+3-2
5 is 4+3-2       5 = 4+3-2       5 =:= 4+3-2
10 is X+3-2      10 = X+3-2      10 =:= X+3-2
1 is sin(pi/2)   1 = sin(pi/2)   1 =:= sin(pi/2)
```

Try `display/1` for showing the actual representation of a term, e.g., `display(4+3-2)`.

V.3 Implement `factorial/2`, e.g., `factorial(3,6)` is true, using a straightforward recursive implementation. Assume that the first parameter is an input and the second one is an output, we usually emphasize this by writing `factorial(+X,-Y)`.

V.4 Check Exercises 3.10 and 3.11 on Flach's slides—the `length` predicate without and with an accumulator.

V.5 Implement `factorial_acc/2` using an accumulator. The point is to move the recursive call in such a way that the result is tail recursive.

Compare the performance of both implementations using `time/1`, see `?-time(factorial(10000,_)).` and `?-time(factorial_acc(10000,_)).`

(Hint: `factorial_acc(X,Y) :- factorial_acc(X,1,Y).`)

V.6 Check the logic programming methodology on Flach's slides. Hence we have `partition/4` and an implementation of the insertion sort algorithm.

V.7 Implement the quicksort algorithm, you can use the implementation of insertion sort algorithm as an inspiration, using `partition/4` and `append/3` (use the one in library `lists`). Clearly, the most important step is the right choice of a pivot, however, use a head of list for simplicity.

V.8 Re-implement the quicksort algorithm using an accumulator instead of `append/3`.