02156 Exercises-12

Jørgen Villadsen

2021

Exercise 1

Write a higher-order program `plist(P,Xs,Ys)` such that `P` is applied elementwise to the list `Xs` and the list `Ys` contains the elements for which the call succeeds.

Example:

```
test(A) :- A > 0.

?- plist(test,[1,2,-3,4,5,0,6],L).

L = [1, 2, 4, 5, 6]

Yes
```

Exercise 2

Extend the SWI-Prolog built-in predicate `sort` with an argument that computes the key for the sorting, for example, if lists are to be sorted by their length (see the first query) rather than by the standard order (see the second query):

```
?- sort([[b],[a,b],[],[a]],A,length).

A = [[], [b], [a], [a, b]]

Yes

?- sort([[b],[a,b],[],[a]],A,=).

A = [[], [a], [a, b], [b]]

Yes
```

Note that the third argument to `sort` must be a binary predicate.

Duplicates must not be merged.

Write a predicate `neg` such that `sort` can be used to sort a list of integers in non-increasing order rather than in non-decreasing order.

Write predicates `fst`, `snd` and `prd` such that `sort` can be used to sort a list of pairs of integers by the first coordinate, by the second coordinate and by the product of the coordinates.

Consider the list of small primes 2, 3, 5 and 7 strictly between 1 and 10. Write a predicate `tst` such that `sort` can be used to sort a list of pairs of integers as follows: first comes the pairs where no coordinates are among the small primes, then comes the pairs where one coordinates are among the small primes and finally comes the pairs where both coordinates are among the small primes, hence for example:

```
?- sort([(3,5),(1,9),(4,2)],A,tst).

A = [ (1, 9), (4, 2), (3, 5)]

Yes
```

Exercise 3

Write a program `testonce(+Goal)` that succeeds iff `Goal` succeeds and in addition it cuts all alternatives of `Goal` (it is already available as a built-in predicate `once` so another name is used in order to test it).

Explain the difference in the following example:

```
t :- true ; true.

p :- t, testonce(t).

q :- t, t, !.

?- p, write(x), nl, fail.
x
x

No

?- q, write(x), nl, fail.
x

No
```

Exercise 4

Use the "Vanilla" meta-interpreter to write a tracer for Prolog in Prolog:

```
?- tracer(member(b,[a,b,c])).
>member(b, [a, b, c])
>member(b, [b, c])
<member(b, [b, c])
<member(b, [a, b, c])

Yes

?- tracer(member(d,[a,b,c])).
>member(d, [a, b, c])
>member(d, [b, c])
>member(d, [c])

No

?- tracer(member(X,[a,b,c])).
>member(a, [a, b, c])
<member(a, [a, b, c])

X = a

Yes
```

The program `tracer` is still just three lines...

Load the predicates `member` and `append` from the file `basic.pl` available on CampusNet (Program files folder in the top folder) — do not use the auto-loaded ones!

Compare the answers to the following queries:

```
?- tracer(append(X,Y,Z)).

?- trace, append(X,Y,Z).
```