Assignment 8

Due: Friday, 1.7.2016, 15:59 via Git

For help, contact alp-staff@lists.iai.uni-bonn.de (staff only) or alp-course@lists.iai.uni-bonn.de (staff and participants).

Start working on the exercises early enough so that you can contact your tutor in time if you have problems.

Submit your implemented predicates as a file named "assignment08/solutions.pl" in the Git repository of your group.

Add a file "assignment08/testRuns.txt" showing the console output of a session in which you test that **each** solution works for the provided input data and some queries that represent sensible test cases.

If no input data is provided in the text of the task, create some sensible input data.
If input data is represented as facts, include them into the "solutions.pl" file and add suitable comments.

Task 1. *Accumulators again* (7 Points)

Write a predicate `sum_square(+IntegerList, ?Sum, ?Square)` that succeeds if

- the first argument is a list of integers provided as input,
- the second argument is the sum of all elements in the list and
- the third argument is the sum of their squares.

For example, `sum_square([1,2,3], 6, 14)` should succeed. Find a solution that is as efficient as possible (using techniques that you have learned so far).

Task 2. *Sorting and Cuts* (8 Points)

Try to understand how the three sorting predicates defined below work. Then

1. (*3 Points*) Name each version by the name of the sorting algorithm that it implements.
2. (*5 Points*) Decide where are good places to use cuts to optimize the different versions. If you find any, explain where you would place the cuts and what would be changed by the cuts.
3. (*1 Points*) Say which version (a, b, c – either with or without cuts) you would prefer and explain the reasons why.

a)
```
sort([],[]).
sort([X],[X]).
sort([H|T1],[H,S|T2]):- sort(T1,[S|T2]), H < S.
sort([H|T1],[S|T3])  :- sort(T1,[S|T2]), not(H<S), sort([H|T2],T3).
```

b)
```
sort1([X|Xs], Ys) :- sort1(Xs, Zs), insert(X, Zs, Ys).
sort1([],[]).

insert(X, [], [X]).
insert(X, [Y|Ys], [Y|Zs]  ):- X > Y, insert(X, Ys, Zs).
insert(X, [Y|Ys], [X,Y|Ys]):- X=<Y.
```

c)
```
sort2([X|Xs], Ys):-
    partition(Xs, X, Littles, Bigs),
    sort2(Littles, Ls),
    sort2(Bigs, Bs),
    append(Ls, [X|Bs], Ys).
sort2([],[]).

partition([X|Xs], Y, [X|Ls], Bs):- X=<Y, partition(Xs, Y, Ls, Bs).
partition([X|Xs], Y, Ls, [X|Bs]):- X >Y, partition(Xs, Y, Ls, Bs).
partition([], Y, [], []).
```

Task 3. *Red Cuts* (5 Points)

Consider the max/3 predicate given in the lecture:

```
max(X,Y,X) :- X >= Y, !.
max(X,Y,Y) .
```

a) (1 Point) Find an invocation of this predicate that behaves incorrectly. Then explain what goes wrong.

b) (2 Points) Can you think of a similar case that would go wrong? Try to state a general problem / principle.

c) (2 Points) Rewrite the predicate so that the problem disappears. From the problem and your solution, try to derive a general rule about how to use cuts.