CSE 463/563 Project E1 (extra credit, due on 11/03/14)

This assignment is about programming in Prolog. In order to complete it, you have to install XSB on your machine (or use a CSE server), and create a text-file name "`ProjectE1.P`"; notice that the extension *must* be a capital P. This file will contain your Prolog code; as usual, you will submit it executing the command `submit_cse563` (or `submit_cse463`) on any CSE server. This is **individual work**. Identical solutions will be considered potential violations of academic integrity. At the beginning of `ProjectE1.P` please insert three Prolog comments, stating your name, your person number, and the phrase **"The submitted solutions are my individual work."** For example:

```
% Niccolo Meneghetti
% 12345678
% The submitted solutions are my individual work.
```

In this assignment you are allowed to declare only facts, Horn clauses and lists. *Do not* use any other Prolog built-in predicate or operator, like the cut ("`!`"), the negation, or any arithmetic operator. Do not use external libraries. You can assume that list elements are integers.

Step 1 (5 pts)

Define a predicate `shorter(L1,L2)`, that is satisfied when either the two lists `L1` and `L2` have the same length, or `L1` is shorter. For example:

```
?- shorter([1], [1,1]) -> yes.
?- shorter([1], [1])   -> yes.
?- shorter([1,1], [1]) -> no.
```

Step 2 (4 pts)

Define a predicate `sorted(LL)`, that is satisfied when the list `LL` contains other lists that are sorted in order of increasing length. For example:

```
?- sorted([[],[1],[1,1],[1,1,1]]) -> yes.
?- sorted([[],[1],[1,1]])         -> yes.
?- sorted([[1],[],[1,1],[1,1,1]]) -> no.
```

HINT: The above two steps can be solved writing two/three lines of code for each predicate. If you are using more rules, you are probably doing it wrong. In the declaration of `sorted/1` the only additional predicate you need is `shorter/2`.

Step 3 (4 pts)

Define a predicate `split(LO,LE,L)`, that is satisfied when the list `LO` contains all the elements with an odd index in the list `L` (i.e. the first element, the third one, the fifth one, and so forth), while `LE` contains all the other elements. For example:

```
?- split([1,3],[2,4],[1,2,3,4]) -> yes.
?- split([1,3],[2],[1,2,3])     -> yes.
```

This will take three lines of code.

Step 4 (4 pts)

Define a predicate `mergesort(M,S1,S2)`. The three parameters are assumed to be lists of lists. When the predicate is satisfied, `M` is obtained by iteratively removing the elements from `S1` and `S2`; at each step the two heads of `S1` and `S2` are compared and the `shorter` one is added at head of `M`. For example:

```
?- mergesort([[1],[1,1],[1,1,1]],[[1],[1,1,1]],[[1,1]])  -> yes.
?- mergesort([[1],[1,1],[1,1,1]],[[1,1]], [[1],[1,1,1]]) -> yes.
?- mergesort([[1,1],[1],[1,1,1]],[[1,1]], [[1],[1,1,1]]) -> no.
?- mergesort([[1],[1,1],[1,1,1]],[[1]],[[1,1]]) -> no.
```

This will take four lines of code. Make sure to use the predicate `shorter/2` defined at step 1, and no other predicate.

Step 5 (3 pts)

Define a predicate `sorted(LS,LU)`, that is satisfied when `LU` is a list of lists and `LS` is obtained by sorting the elements of `LU` in order of increasing size.

```
?- sorted([[],[1],[1,1],[1,1,1]], [[1,1,1],[1],[1,1],[]]) -> yes.
?- sorted([[1],[],[1,1],[1,1,1]], [[1,1,1],[1],[1,1],[]]) -> no.
?- sorted([],[])                                           -> yes.
```

Final testing

Download from Piazza the file "`queries.P`", and use it for testing your solution. The file defines five queries, one for each step. You can run them entering one of the following command from the console:

```
xsb -e "['ProjectE1.P'],['queries.P'],q1."
xsb -e "['ProjectE1.P'],['queries.P'],q2."
...
xsb -e "['ProjectE1.P'],['queries.P'],q5."
```

The output is either `SUCCESS` or `FAILURE`, and will determine how many points you receive for this assignment. Make sure to use the correct names for the predicates and to handle recursions properly.