Consider the Regular Grammar G:
1 S ---> a
2 X ---> b
3 S ---> aS
4 S ---> aX

Here is DB representing the rules for the grammar:

1* s( [a], [1] ).
2  x( [b], [2] ).
3* s( [a| L1], [3 | R] ) :- length(L1, N), N > 0, s(L1, R).
4  s( [a| L1], [4 | R] ) :- length(L1, N), N > 0, x(L1, R).

Here is a query which, given a string generated by the grammar G, gives a sequence (list) of rules which generated that string. following the query is a trace that shows the result and how it is developed.

?- s([a, a, b], R)= [3, 4, 2] ). success (Returned values for R are green)
  1 s( [a], [1] ). fail
  2 x( [b], [2] ). fail
  3 s( [a| L1], [3 | R= [4,2] ] :- length([a, b], 2), 2 > 0,
      \ s([a, b], R). success
      \  1 s( [a], [1] ). fail
      \  2 x( [b], [2] ). fail
      \  3 s( [a| L1], [3 | R ] :- length([ b ] , 1), 1 > 0,
             \     s([b], R). All Fail Backup
             \      1 s( [a], [1] ). fail
             \      2 x( [b], [2] ). fail
             \      3 s( [a| L1], [3 | R ] :- length(L1, N), N > 0, s(L1, R).fail
             \       4 s( [a| L1], [4 | R] ) :- length(L1, N), N > 0, x(L1, R).fail
             \ All Fail Backup
             \  4 s( [a| L1], [4 | R=2 ] :- length(L1, 1), 1 > 0,
                 \   x([b], R=2 )).success
                 \    1 s( [a], [1] ). fail
                 \    2 x( [b], [2] ). success

1* [From NTS S the TS a can be derived with rule 1]
3* [if a length N (N>0) list, L1, is deriveable from S with rule list R , then derivable from S is the list [a|L1], with rule list [3|R]]
Questions: 80% for 1 100% for 1 and 2 105% for 1,2, and 3.

1) Design a running DB for the following grammar, G1, which allows making the same type of query as shown on previous page for G.

2) Design a second running DB which when queried as above returns the the input sequence which was to be parsed with words added as illustrated below. If the query is ?- s([a,a,*,a,a,a,+,a,a],Z). the response is

   Z = [[startopen a],[a],[close,*,open],[a],[a],[a],[close,+open],[a],[a,closend]]
   [representing (a,a,)*,(a,a,a,)+,(a,a)]

3) Add to the DB a statement smash(Z,Q) :- ........ which can be used to remove the inner list brackets in a list of lists like Z in 2 above. So for that Z, for query ?- smash(Z,Q) the response is

   Q = [startopen a,a,close,*,open,a,a,a,close,+open,a,a,closend],