Prolog Lists: Work with your partner(s) to understand and write Prolog relations which manipulate lists.

Follow the instructions:

1. Login to phoenix. Download the ch7_1.pl file from goucherLearn. This file contains a relation blockList which tests for a legal list of blocks. Lists can be written in different ways. Test each of the following to see if it is a legal list of blocks and explain each result.

   a. blockList([]).

   b. blockList([b2]).

   c. blockList([1,2,3]).

   d. blockList([b2,b3,b1]).

   e. blockList([b2|[b3,b1]]).

   f. blockList([b2,b3|b1]).

   g. blockList([b2,b3 | [b1]]).

   h. blockList([b2,b3 | []]).

2. The member relation tests if a particular value is contained within a list. It sure makes writing the relation uniq_blocks so much easier than what we were doing before and it works with *any* number of blocks! Do you understand how uniq_blocks works? Try it out.

3. Download the ch7_2.pl file from goucherLearn. This file contains a list version of the blocks world on p66 of your text. Each stack is now represented as a list of blocks. Then we have a list of stacks. This is defined in the relation *scene*.

   The *before* relation uses the relation append(A,B,C) which determines if lists A and B glued together comprise list C.
   before(X,Y,L) :- append(Z,[Y|_],L) and append(_,[X|_],Z).

   Give a value for Z which would make the relation before(1,2,[3,1,4,5,2,6]) hold.

4. The *left* relation uses *before* and *member* to determine whether one block appears to left of another in the scene. Give values of Stack1 and Stack2 which make the relation left(b1,b5) hold.

5. Write the relation just_before(X,Y,L) as described on p151, exercise #4.

6. Write the relation on(X,Y) as described on p151, exercise #5.

7. Write the relation intersect(X,Y) as described on p151, exercise#6d.