Comp 333 Project 4 (40 pts) Fall 2018

**Due Tuesday Dec 11 in class (No Canvas submission)**

**DIRECTIONS:**

1. Turn in a printout of your Prolog source file (project4.pl) with your Name, Project #, Comp 333( 8AM or 11AM class ) embedded at the top of the file. Add a cover page.
2. Copy and paste the results of testing your Prolog functions as comments inside the project4.pl file. Number each problem very clearly. Put the problems in numeric order. For each problem the layout should look list this:

% ---------------------------------------------

%Problem 1

%Comment if code missing or if it does not run

%Code for problem 1

%Test case query and results for problem 1 ( copied and pasted from the SWI Prolog interpreter)

% ----------------------------------------------------------------

1. Test Cases for each problem will be posted on Canvas on or before Dec 4.
2. Do not submit Project 4 to Canvas. Submit only hard copy in class on Dec 11. No late projects accepted.

**Project 4 PROBLEMS**

1. (5 pts) Use append to write the following predicates about a list L.

* endWith(L,X). Does list L end with an X?
* beginWith(L, A) Does list L begin with the A?
* split(L, Z, L1, L2). Split L at symbol Z. Return the split. Put the symbol Z at the beginning of list L2. It should work like this:

?-split( [3,5,6,8,10,4,5], 10, Left, Right]

Left = [3,5,6,8]

Right = [10,4,5]

1. (5 pts) Write a recursive Prolog predicate maxList(L, R) to find the maximum in a list of numbers. Here L is the list and R is the maximum value in the list. It should work like this

?- maxList( [ 3,1,8,2, 5], R).

R = 8

?- maxList( [ 7], Z).

Z = 7.

?- maxList([ ], A).

false

1. (5 pts) Write a Prolog predicates about a list L

* all\_ab(L). L is a list of only a’s and b’s, with at least one a or b.
* all\_a(L), L is a list of only a’s with at least one a
* all\_b(L). L is a list of only b’s with at least one b
* aplus\_bplus(L). L is a list of a’s followed by b’s with least one a and at least one b.

1. (5 pts) Write a recursive Prolog predicate insertIt(Item, L,R) to insert an Item into list L. The list R contains the result. Repeated calls to insertIt should insert the item into all possible locations.

Hint:

insertIt(X, [], R ) :- \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

insertIt( X, [H| T, R] :- \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ //insert X into head of list

insertIt(X, [H|T] , R ) :- \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ //insert X into rest of list

?- insertIt( a, [ d,o,g] , R).

R = [a,d,o,g];

R = [d,a,o,g];

R= [d,o,a,g];

R = [d,o,g,a];

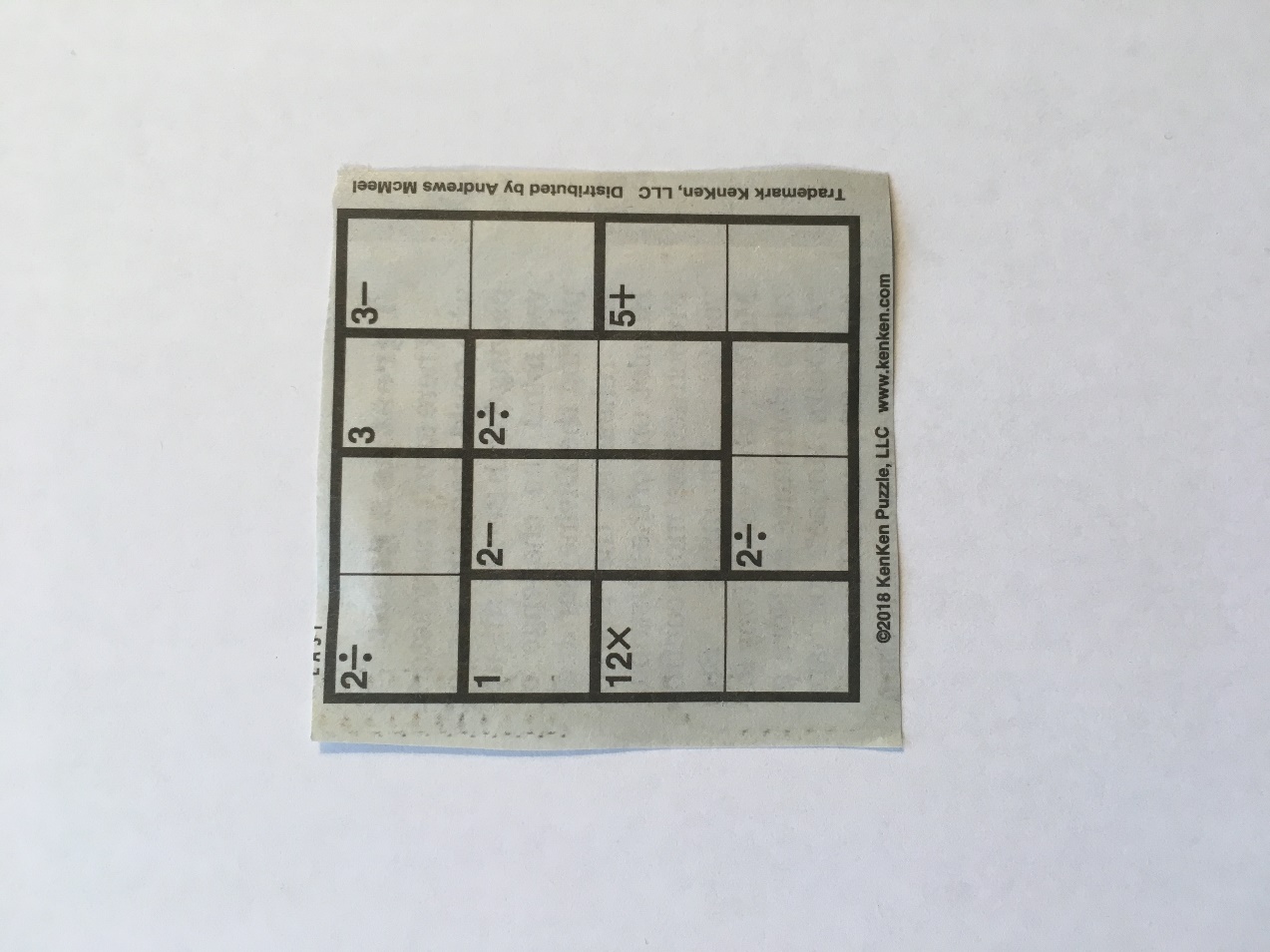
R = false.

1. (10 pts) Use Prolog to solve the 4x4 KENKEN problem describe on the next page. Label the squares X1,X2, …, X16 from top to bottom, from left to right. Then (required)
   1. Use the Prolog permutation predicate to make sure that each row uses all the numbers 1- 4 exactly once.
   2. Check that the columns have no duplicates. Use the predicate is\_set to determine if a list has NO duplicates.
   3. Check that the cages meet the cage requirements
   4. If your problem takes too long to run, you can reorder or intermingle the steps in a,b,c.
   5. Your solution should be easily and obviously adaptable to any 4x4 KENKEN problem with the same cages but different targets and operations. **As part of solving the problem, describe how you would adapt your solution to a different KEKKEN problem. Include the description as a comment in your project4.pl file.**

<SEE NEXT PAGE for description of the KENKEN Problem>

Description of 4x4 KENKEN problem.

Every box will contain an integer between 1 and 4. Do not repeat a number in any row or column. The numbers in each heavily lined set of squares ( cage) must combine to produce a target number found in the top left corner of the cage using the mathematical operation indicated. A number can be repeated in a cage as long as it is not in the same row or column.



1. (10 pts) Challenge: Write a Prolog program to solve the simplified Knapsack Problem: Given a list of unique positive integer weights Weights and a knapsack capacity W, find the subset of the weights in Weights whose sum <= W and for which the sum is a maximum. Find all solutions.

Examples:

Weights = [5, 4, 10, 12, 3, 2], W = 27

Solutions: OptVal = 27, OptSet = [5, 10, 12] ;

OptVal = 27, OptSet = [10, 12, 3, 2] ;

Weights = [5, 4, 10, 12, 3, 2], W = 35

Solutions: OptVal = 34, OptSet = [5, 4, 10, 12, 3] ;

Hint: Use the Subset Sum Problem solution from the Prolog power point slides.