**Q1. Define the relation max( X, Y, Max) so that Max is the greater of two numbers X and y.**

**Sol 1:**

max(X, Y, Max):-

(

X > Y->

Max is X

;

Max is Y

).

**Query: ?-** max(X, Y, Max).

**Q2. Define the predicate maxlist(List, Max) so that Max is the greatest number in the list of numbers List.**

**Sol 2 :**

maxlist([], R, R).

maxlist([X|Xs], WK, R):- X > WK, maxlist (Xs, X, R).

maxlist([X|Xs], WK, R):- X =< WK, maxlist(Xs, WK, R).

maxlist([X|Xs], R):- maxlist(Xs, X, R).

**Query: ?-** maxlist ([1, 2, 3, 4, 45, 6, 7, 8], Max).

**3 Define the predicate sumlist( List, Sum) so that Sum is the sum of a given list of numbers List.**

**Sol 3:**

sumlist([], 0).

sumlist([Head|Tail], Sum):- sumlist(Tail, Temp), Sum is Temp + Head.

**Query: ?-** sumlist([1, 2, 3, 4, 5], Sum).

**4. The following relation classifies numbers in to three classes positive, zero, and negative: class( Number, positive) :- Number > 0.**

**class( 0, zero).**

**class( Number, negative) :- Number < 0.**

**Define this procedure in a more efficient way using cuts.**

**Sol 4:**

class(Number, positive) :- Number > 0, !.

class(0, zero) :- !.

class(Number, negative) :- Number < 0, !.

**5. Define the procedure split( Numbers, Positives, Negatives) which splits a list of numbers into two lists: positive ones ( including zero) and negative ones. For example, split([ 3,-1,0,5,-2],[3 ,0,5],[ -1,-21])**

**Propose two versions: one with a cut and one without.**

**Sol 5:**

**Using Cut**

split([], [], []).

split([HP | TL], [HP | TP], N):- HP >= 0, !, split(TL, TP, N).

split([HN | TL], P, [HN | TN]):- HN < 0, !, split(TL, P, TN).

**Without Cut**

split([], [], []).

split([Head|Tail], [Head|List1], List2):- Head>=0, split(Tail, List1, List2).

split([Head|Tail], List1, [Head|List2]):- Head<0, split(Tail, List1, List2).

**Query:** split([1, 2, -5, 7, 55, 0, -4], P, N).

**6. Consider this prolog program and answer the query for a cut-free program and this program with cut. Give your answer by drawing proof tree.**

**p(X):- a(X).**

**p(X):- b(X), c(X), !, d(X), e(X).**

**p(X):- f(X).**

**a(1).**

**b(1).**

**b(2).**

**c(1).**

**c(2).**

**d(2).**

**e(2).**

**f(3)**

**?- p(X).**

**Sol 6:**

**Output 1) ?-** X=1;false

**2)**

p(X):- a(X).

p(X):- b(X),c(X),d(X),e(X).

p(X):- f(X).

a(1).

b(1).

b(2).

c(1).

c(2).

d(2).

e(2).

f(3).

**Output:** ?- p(X):

X = 1;

X = 2;

X = 3;

**7. Please refer slide no 45 in lecture slide dated 5/5/2020 (3 DCG and CLPFD.pptx) to answer these questions:**

**i) Whether this same is sufficient to answer queries for both way translation? If not, modify this program to do it.**

**ii) Explain the purpose of phrase predicate in this program.**

**iii) Explain the purpose of cut operator in this program.**

**Sol 7:**

i) Yes, it is performing both way translation.

ii) The SWI-Prolog implementation of phrase/3 verifies that the List and Rest arguments are unbound, bound to the empty list or a list cons cell. Other values raise a type error. The predicate call\_dcg/3 is provided to use grammar rules with terms that are not lists.

**8. Write a cryptarithmetic puzzle solver using clpfd for this puzzle: HAIKU+SUSHI=KIMONO.**

**Sol 8:**

:- use\_module(library(clpfd)).

puzzle([H,A,I,K,U] + [S,U,S,H,I] =[K,I,M,O,N,O]) :-

Vars = [H,A,I,K,U,S,M,O,N],

Vars ins 0..9,

all\_different(Vars),

H\*10000 + A\*1000 + I\*100 + K\*10 + U +

S\*10000 + U\*1000 + S\*100 + H\*10 + I #=

K\*100000 + I\*10000 + M\*1000 + O\*100 + N\*10 + O,

H #\= 0, S #\= 0, K #\= 0.