% Question 1

%

weirdsum(Numbers,Weird) :-

include(greater\_than\_five,Numbers,Gtf),

include(less\_then\_two,Numbers,Ltt),

maplist(sqr,Gtf,SqrGtf),

maplist(abs,Ltt,AbsLtt),

sumlist(SqrGtf,SmSqr),

sumlist(AbsLtt,AbsSum),

Weird is SmSqr - AbsSum.

greater\_than\_five(Number) :-

Number >= 5.

less\_then\_two(Number) :-

Number =< 2.

sqr(X,Y) :-

Y is X \* X.

% Question 2

%

% QUESTION 2:

% same\_name(Person1, Person2) :- Both persons are sharing the same family name

%

% Person1 and Person2 have the same father

same\_name(Person1, Person2) :-

parent(Parent, Person1),

parent(Parent, Person2),

male(Parent),

Person1 \= Person2.

% Person1 is the father of Person2

same\_name(Person1, Person2) :-

parent(Person1, Person2),

male(Person1),

Person1 \= Person2.

% Person2 is the father of Person1

same\_name(Person1, Person2) :-

parent(Person2, Person1),

male(Person2),

Person1 \= Person2.

% Person1 is the father of X, who

% is the father of Person2

same\_name(Person1, Person2) :-

parent(Person1, X),

parent(X, Person2),

male(Person1),

male(X),

Person1 \= Person2.

% Person2 is the father of X, who

% is the father of Person1

same\_name(Person1, Person2) :-

parent(Person2, X),

parent(X, Person1),

male(Person2),

male(X),

Person1 \= Person2.

% Question 3a

%

log\_table([], []).

log\_table([Head|Tail], ResultList) :-

log\_table(Tail, SubList),

X is log(Head),

ResultList = [[Head, X]|SubList].

% Question 3b

%

function\_table([], \_, []).

function\_table([Head|Tail], Function, ResultList) :-

function\_table(Tail, Function, SubResult),

X =.. [Function, Head],

Y is X,

ResultList = [[Head|Y]|SubResult].

% Question 4

%

paruns([],[]).

paruns([Head | Tail],RunList):-

paruns(Tail,RestList),

padd(Head, RestList, RunList).

padd(Head, RestList, RunList):-

RestList == [],

RunList = [[Head]].

padd(Head, RestList, RunList):-

[[H | T] | Rest] = RestList,

H mod 2 =:= Head mod 2,

RunList = [[Head | [H | T]] | Rest].

padd(Head,RestList,RunList):-

[[H | T] | Rest] = RestList,

H mod 2 =\= Head mod 2,

RunList = [[Head]|RestList].

% Question 5

% Binds Eval to the result of evaluating the expression-tree

% Tree, with the variable z set equal to the specified Value.

%

is\_heap(empty).

is\_heap(tree(L,K,R)):-

is\_heap(L),

is\_heap(R),

(L = tree(\_, LK, \_) \*-> K @=< LK; true),

(R = tree(\_, RK, \_) \*-> K @=< RK; true).

% Question: 5B

%

height\_if\_balanced(empty, 0).

%height\_if\_balanced(tree(empty, \_, empty), 1).

%height\_if\_balanced(tree(Left, \_, empty), HiB) :-

% height\_if\_balanced(Left, SubCount),

% HiB is SubCount + 1.

%height\_if\_balanced(tree(empty, \_, Right), HiB) :-

% height\_if\_balanced(Right, SubCount),

% HiB is SubCount + 1.

height\_if\_balanced(tree(L, \_, R), HiB) :-

height\_if\_balanced(L, L\_height),

height\_if\_balanced(R, R\_height),

1 < abs(L\_height - R\_height),

HiB is -1.

height\_if\_balanced(tree(L, \_, R), HiB) :-

height\_if\_balanced(L, L\_height),

height\_if\_balanced(R, R\_height),

1 >= abs(L\_height - R\_height),

max(L\_height, R\_height, Longest),

HiB is Longest + 1.

% max(A, B, C) binds C to the larger of A and B.

%

max(A, B, A) :-

A > B.

max(A, B, B) :-

A =< B.