```
% Problem 1: Pasaran Days
% The pasaran days are: pahing, pon, wage, kliwon, legi, respectively.

day(Start,1,Start). % the base case.

% we have five recursive case as follows:

day(Start,Day,pon):- Day > 1, Day1 is Day  - 1, day(Start,Day1,pahing).
day(Start,Day,wage):- Day > 1, Day1 is Day - 1, day(Start,Day1,pon).
day(Start,Day,kliwon):- Day > 1, Day1 is Day - 1, day(Start,Day1,wage).
day(Start,Day,legi):- Day > 1, Day1 is Day - 1, day(Start,Day1,kliwon).
day(Start,Day,pahing):- Day > 1, Day1 is Day - 1, day(Start,Day1,legi).

% Problem 2: Triangles
% Part 1: inverted triangle

% star(N) returns a string of N stars
star(1):- write('*'). % base case
star(N):- N > 1, write('*'), N1 is N - 1, star(N1). % recursive case

/* inverttri(N) yields an inverted triangle of height N
as in the test cases. The predicate returns N lines of
strings, the string on line i contains i characters of stars. */

inverttri(N):-
    N > 0,
    star(N), nl, % writing N stars
    N1 is N - 1,
    inverttri(N1). % calling inverttri(N-1)

% Part 2
/* startri(N) yields a triangle of height N as in the the test cases.
The predicate returns N lines of strings, the string on line i contains i
characters of stars. */

startri(1):- star(1). % base case
startri(N):-
    N > 1,
    N1 is N - 1,
    startri(N1),nl, % calling startri(N-1)
    star(N). % writing N stars

% Part 3
/* tristar(N) yields a triangle of height N as in the test case.
The predicate returns N lines of strings, the string on line i
contains i characters of stars, preceded by N - i blank spaces. */

% space(N) returns a string of N blank spaces
space(0):- write(''). % space(0) returns no space
space(N):- N > 0, write(' '), N1 is N - 1, space(N1).

/* tristar(N) returns N lines of stars, a line i for 1 =< i =< N
contains i stars, each line i has N - i leading blank spaces,
it is easier to work with a predicate tristar(N,TotalStars),
since the total number of spaces and the total number of stars in each line
is always equal maximum number of stars */

tristar(N,MaxStars):-
    N > 0,
    Space is N - 1, Star is MaxStars - Space,
    space(Space),star(Star), nl,
    tristar(Space,MaxStars).

tristar(N):- tristar(N,N).



% Problem 3: The Long Travels
byCar(auckland,hamilton).
byCar(hamilton,raglan).
byCar(valmont,saarbruecken).
byCar(valmont,metz).

byTrain(metz,frankfurt).
byTrain(saarbruecken,frankfurt).
byTrain(metz,paris).
byTrain(saarbruecken,paris).

byPlane(frankfurt,bangkok).
byPlane(frankfurt,singapore).
byPlane(paris,losAngeles).
byPlane(bangkok,auckland).
byPlane(losAngeles,auckland).

% defining auxiliary predicate travel/2
travel(X,Y):- byCar(X,Y).
travel(X,Y):- byTrain(X,Y).
travel(X,Y):- byPlane(X,Y).

travelable(X,Y):- travel(X,Y). % base case
travelable(X,Y):- travel(X,Z),travelable(Z,Y). % recursive case

% defining auxiliary predicate travel1/3, with go/2 as a functor.
travel1(X,Y,go(X,Y)):- byCar(X,Y).
travel1(X,Y,go(X,Y)):- byTrain(X,Y).
travel1(X,Y,go(X,Y)):- byPlane(X,Y).

travelwhere(X,Y,go(X,Y)):- travel1(X,Y,go(X,Y)). % base case
% if we see the pattern, we need to define go/3 recursively as follows:
travelwhere(X,Y,go(X,A,B)):- travel1(X,A,go(X,A)), travelwhere(A,Y,B).

% defining auxiliary predicate travel2/3, with go/3 as a functor
travel2(X,Y,go(X,Y,car)):- byCar(X,Y).
travel2(X,Y,go(X,Y,train)):- byTrain(X,Y).
travel2(X,Y,go(X,Y,plane)):- byPlane(X,Y).

travelhow(X,Y,go(X,Y,Type)):- travel2(X,Y,go(X,Y,Type)). % base case.
```

```
% if we see the pattern, we need to define go/4 recursively as follows:
travelhow(X,Y,go(X,A,Type,B)):- travel2(X,A,go(X,A,Type)), travelhow(A,Y,B).

% Problem 5: Greatest Common Divisor
/*version 1: exploiting the recursion gcd(a,b) = gcd (b,a mod b),
uncomment to try
gcd(0,0,_):- write("gcd error"). % handling exception when both numbers are zero
gcd(A,0,A):- A =\= 0. % gcd(A,0) = A for nonzero A
gcd(0,A,A):- A =\= 0. % gcd(0,A) = A for nonzero A
gcd(A,A,A):- A =\= 0. % gcd(A,A) = A for nonzero A
gcd(A,B,Gcd):- % if 0 < A < B, then gcd(A,B) = gcd(A,B mod A)
    0 < A, A < B,
    C is B mod A,
    gcd(A,C,Gcd).
gcd(A,B,Gcd):-  % if 0 < B < A, then gcd(A,B) = gcd(B, A mod B)
    0 < B, B < A,
    C is A mod B,
    gcd(B,C,Gcd).
*/

% version 2: using the hint
gcd(0,0,_):- write("gcd error"). % handling exception when both numbers are zero
gcd(A,0,A):- A =\=0. % gcd(A,0) = A for nonzero A
gcd(0,A,A):- A =\=0. % gcd(0,A) = A for nonzero A
gcd(A,A,A):- A =\=0. % gcd(A,A) = A for nonzero A
gcd(A,B,Gcd):-  % if 0 < A < B, then gcd(A,B) = gcd (A,B-A) (3rd case).
    0 < A,
    A < B,
    C is B-A,
    gcd(A,C,Gcd).
gcd(A,B,Gcd):- % If 0 < B < A, then gcd(A,B) = gcd(A-B,B) (4th case).
    0 < B,
    B < A,
    gcd(B,A,Gcd).

% Problem 5: Favorite Meals
:- op(650, xfx, suka).
alia suka mie.
alia suka bakso.
alia suka rendang.
alia suka eskrim.

bambang suka bakso.
bambang suka sate.
bambang suka coklat.
bambang suka eskrim.

caca suka sate.
caca suka mie.
caca suka bakso.
caca suka coklat.

dani suka bakso.
dani suka sate.
dani suka rendang.
dani suka eskrim.

:- op(600, xfy, dan).
% The distributivity of dan.
A suka B dan C:- A suka B, A suka C.
%===================================

% Problem 6: Happy Pi Day!
% approximating pi with reciprocals sum
% computing the sum of N reciprocals of squares of natural numbers

sumreciprocal(1,1).
sumreciprocal(N,SumrecN):-
    N > 1,
    M is N - 1,
    sumreciprocal(M,SumrecM),
    SumrecN is SumrecM + 1 / (N ^2).

% approximation of pi
approxpi(N,Approx):-
    sumreciprocal(N,SumrecN),
    Approx is sqrt(6 * SumrecN).
```