```prolog
% Problem 1
% The first version exploits the built-in function max(M,N).
/* The predicate X is max(M,N) succeeds whenever M and N are numbers
and it unifies X to the maximum value of M and N.*/
max3numbers(X,Y,Z):-
    A is max(X,Y), % A is the maximum of X and Y.
    B is max(A,Z), % Mathematically, B is equal to max(max(X,Y),Z).
    write(B).

/* Another solution, remove the comment for trial.
max3numbers(X,Y,Z):- X >= Y, X >= Z, write(X). % X is the largest number.
max3numbers(X,Y,Z):- Y >= X, Y >= Z, write(Y). % Y is the largest number.
max3numbers(X,Y,Z):- Z >= X, Z >= Y, write(Z). % Z is the largest number.
*/

% Problem 2.

% The base case.
mypromise(1):-
    write("I will study hard for the midterm.").

% The recursive case.
mypromise(N):-
    N>1, % Lower bound to avoid infinite recursive call.
    write("I will study hard for the midterm."),
    nl, % Move to a new line.
    M is N-1, % Update the value of M.
    mypromise(M). % Calling mypromise/1 recursively.

% Problem 3.

% The base case.
factorial(0,1). %0! = 1.
factorial(1,1). %11 = 1.

% The recursive case.
factorial(N,FactN):-
    N > 1, % Lower bound to avoid infinite recursive call.
    M is N-1, % Decreasing the value of N to N-1.
    factorial(M,FactM), % Calling factorial/2 recursively.
    FactN is FactM * N. % Updating the result.

/* As we all know in discrete mathematics, factorial function can be
defined recursively, namely: 0! = 1, 1! = 1, and n! = n (n-1)! for n > 1.*/

% Problem 4.
% The base case.
sumcube(0,0).
sumcube(1,1).

% The recursive case.
sumcube(N,SumN):-
    N > 1, % Lower bound to avoid infinite recursive call.
```

```
    M is N-1, % Decreasing the value of N to N-1.
    sumcube(M,SumM), % Calling sumcube/2 recursively.
    SumN is SumM + N^3. % Updating the result.

/* Suppose S(n) = sum of cube from 1 up to n, i.e.:
S(n) = 1^3 + 2^3 + 3^3 +...+ n^3.
We have S(n+1) = sum of cube from 1 up to n+1, i.e.:
S(n+1) = 1^3 + 2^3 + 3^3 + ... + n^3 + (n+1)^3.
It is easy to see that:
S(n+1) = S(n) + (n+1)^3, hence we get the recurrence:
S(0) = 0, S(1) = 1, and S(n) = S(n-1) + n^3 for all n > 1.
*/

% Problem 5.
% The knowledge base.
directTrain(forbach,saarbruecken).
directTrain(freyming,forbach).
directTrain(fahlquemont,stAvold).
directTrain(stAvold,forbach).
directTrain(saarbruecken,dudweiler).
directTrain(metz,fahlquemont).
directTrain(nancy,metz).

% The base case.
travelBetween(A,B):- directTrain(A,B).
% Handling symmetry for the base case.
travelBetween(A,B):- directTrain(B,A).

% The recursive case.
travelBetween(A,B):- directTrain(A,C), travelBetween(C,B).
% Handling symmetry for the recursive case.
travelBetween(A,B):- directTrain(B,C), travelBetween(C,A).
```