Ce document contient juste les codes pour présentation. **J’ai aussi mis les codes en format txt et pl** **dans le même zip** pour faciliter la correction.

Question 1:

a)

%insertion in the list

insert(A,L,[A|L]).

insert(A,[X|L], [LL|X]) :- insert(A,L,LL).

%basic cases

greater([A],[B],E) :- A > B, T=E, insert('#t',T,E).

greater([A],[B],E) :- A < B,T=E, insert('#f',T,E).

% lists with more elements

greater([A|B],[C|D],E) :- A > C,insert('#t',N,E), greater(B,D,N).

greater([A|B],[C|D],E) :- A < C,insert('#f',N,E) ,greater(B,D,N).

b)

%insertion in the list

insert(A,L,[A|L]).

insert(A,[X|L], [LL|X]) :- insert(A,L,LL).

%basic cases

greater([A],[B],E) :- A > B, T=E, insert('#t',T,E).

greater([A],[B],E) :- A < B,T=E, insert('#f',T,E).

% lists with more elements

greater([A|B],[C|D],E) :- A > C,insert('#t',N,E), greater(B,D,N).

greater([A|B],[C|D],E) :- A < C,insert('#f',N,E) ,greater(B,D,N).

%lists with differents size 1 elements

greater([A],[],E) :- T=E, insert('#t',T,E).

greater([],[C],E) :- T=E, insert('#f',T,E).

% Lists with differents size more than 1 element

greater([A|B],[],E) :- insert('#t',N,E), greater(B,[],N).

greater([],[C|D],E) :- insert('#f',N,E), greater([],D,N).

Question 2:

a)

% Question 2 – a : Determiner le nombre de vitres adjacentes

% trouver la longueur de la liste

list\_size([], 0 ).

list\_size([\_|Xs], L ) :- list\_size(Xs,N) , L is N+1 .

% predicat

% trouver les elements et les compter

neighbors(G,N):- integer(N), bagof(B,adj(G,B),X), bagof(B,adj(B,G),Y), L =[X|Y], list\_size(L,N).

b)

% Question 2 – b :

% Écrire le prédicat permettant de trouver

% la liste des vitres adjacentes à une certaine vitre.

neighbors(G,L) :- bagof(B,adj(G,B),X), bagof(B,adj(B,G),Y), L =[X|Y].

c)

% Question 2- c : c) Écrire le prédicat permettant de

% trouver si 2 vitres ont un voisin en commun.

same\_neighbor(G,I,N):- adj(G,N) , adj(I,N).

same\_neighbor(G,I,N):- adj(N,G) , adj(N,I).

same\_neighbor(G,I,N):- adj(G,N) , adj(N,I).

same\_neighbor(G,I,N):- adj(N,G) , adj(I,N).

Question 3 :

% trouver la taille d'une liste

list\_size([], 0 ).

list\_size([\_|Xs], L ) :- list\_size(Xs,N) , L is N+1 .

% Obtenir les tailles des listes de voteurs par candidats

voters\_list\_size(A,N):- bagof(G,vote(G,A),L),list\_size(L,N).

% 1 candidat

elect([A],E):- E= A.

% aucun candidat

elect([],personne).

% Plusieurs candidats

%% Comparer deux elements consecutifs, trouver le vainqueur

% puis le comparer avec la suite de la liste

elect([A|B],E):- [T|Q] = B , voters\_list\_size(A,X), voters\_list\_size(T,Y), X>Y, E = Z, elect([A|Q],Z).

elect([A|B],E):- [T|Q] = B , voters\_list\_size(A,X), voters\_list\_size(T,Y), X<Y, E = Z, elect(B,Z).

Question 4 :

% Reponse

% Trouver toutes les notes obtenues lors de l'evaluation

marks(Course,Evaluation,L):- findall(N,mark(Course,\_,Evaluation,N),L).

% Trouver la taille de la liste contenant les notes

list\_size([], 0 ).

list\_size([\_|Xs], L ) :- list\_size(Xs,N) , L is N+1 .

% Caluler la somme des elements de la liste

sommer([], 0) :- !.

sommer([T|Q], Somme) :- sommer(Q, S),Somme is T + S.

% calculer la moyenne de l'evaluation

averageMark(Course,Evaluation,Avg) :- marks(Course,Evaluation,L),list\_size(L,N) ,sommer(L,Somme),evaluation(Course,Evaluation,Max),Avg is ((Somme/N)/Max)\*100.