Logo

Description automatically generated with medium confidence

**Compte rendu TP 4 :**

Programmation Logique

**Travail réalisé par :**

Mohamed Aziz Bellaaj

Louay Badri

Talel Ayed

**Groupe :** GL 3/1

Base.pl :

|  |
| --- |
| % Travail fait par Mohamed Aziz Bellaaj, Louay Badri et Talel Ayed  %\* Exercice 1  %! 1. Déclarez les relations rocher(X,Y) et arbre(X,Y) qui retourneront vrai si la case (X,Y) est encombrée par un rocher (respectivement par un arbre).  :- dynamic case/2.  :- dynamic arbre/2.  :- dynamic rocher/2.  %! 2. Déclarez la relation vache(X, Y, Race, Etat) qui retourne vrai lorsqu’une vache de race Race est sur la case (X,Y). L'Etat de la vache est vivante ou zombie. Les races possibles sont brune, simmental, alpine\_herens.  race(brune).  race(simmental).  race(alpine\_herens).  etat(vivante).  etat(zombie).  %! Declare dynamic relation vache  :- dynamic vache/4.  vache(X, Y, Race, Etat):-      case(X, Y),      race(Race),      etat(Etat).  vache(X, Y):- vache(X, Y, \_, \_).  %! 3. Déclarez  la  relation  dimitri(X,  Y)  qui  donne  la  position  de Dimitri.  :- dynamic dimitri/2.  %! 4. Définissez les faits largeur(X) et hauteur(Y) qui donnent la largeur et la longueur du plateau de jeu (choisissez des valeurs).  largeur(8).  hauteur(8).  %! 5. Définissez les faits nombre\_rochers(N), nombre\_arbres(N), nombre\_vaches(Race, N) qui donnent le nombre de rochers, d’arbres et de vaches de chaque race sur le plateau de jeu (choisissez des valeurs).  nombre\_rochers(8).  nombre\_arbres(5).  nombre\_vaches(brune, 5).  nombre\_vaches(simmental, 8).  nombre\_vaches(alpine\_herens, 10).  %\* Exercice 2  %! 1. Écrivez  la  règle  occupe(X,Y)  qui  est  vrai  si  et  seulement si  la  case  (X,Y)  est  occupée  par un arbre, un rocher, une vache ou Dimitri.  occupe(X, Y) :- arbre(X, Y); rocher(X, Y); vache(X, Y); dimitri(X, Y).  %! 2. Écrivez la règle libre(X,Y) qui retourne dans X et Y les coordonnées d’une case libre, c’est-à-dire n’ayant ni rocher, ni arbre, ni vache, ni Dimitri.  %? la logique est de chercher une place random qui n est pas occupe  libre(X,Y) :-      largeur(L),      hauteur(H),      repeat,      X is random(L),      Y is random(H),      (not(occupe(X, Y))),!.  %! 3. Écrivez les règles placer\_rochers(N), placer\_arbres(N), placer\_vaches(Race, N) qui placent N rochers, arbres ou vaches sur le plateau de jeu.    %? Condition d'arret d'insertion d'arbre  placer\_arbres(0).  %? L'insertion de N arbres  placer\_arbres(N) :-      N>0,      libre(X, Y),      assert(arbre(X, Y)),      N1 is N-1 ,      placer\_arbres(N1).  %? Condition d'arret d'insertion d'rocher  placer\_rochers(0).  %? L'insertion de N rocher  placer\_rochers(N) :-      N>0,      libre(X, Y),      assert(rocher(X, Y)),      N1 is N-1 ,      placer\_rochers(N1).  %? Condition d'arret d'insertion de vache  placer\_vaches(\_, 0).  %? L'insertion de N vaches  placer\_vaches(R, N) :-      N>0,      libre(X, Y),      assert(vache(X, Y, R, vivante)),      N1 is N-1 ,      placer\_vaches(R, N1).  %? L'insertion de Dimitri  placer\_dimitri :-      libre(X, Y),      assert(dimitri(X, Y)).  %! 4. Écrivez  la  règle  vaches(L)  qui  retourne  dans  L  la  liste  des  positions  occupées  par  des vaches. Indications : Pensez à utiliser bagof ou findall.  %? Exemple de resultat de L [[1, 2], [4, 5], [3, 4]]  vaches(L) :-      findall([X, Y], vache(X, Y, \_, \_), L).  %! 5.  Écrivez la règle creer\_zombie qui sélectionne aléatoirement une vache et la transforme en zombie.  %? nth0 prend 3 parametres nth0(X, L, R) et retourne X est l'indice ,L est la liste , R est la result qui est l'element de la Liste L d'indice X  creer\_zombie :-      vaches(L),      length(L, Len), X is random(Len),      nth0(X, L, Val), nth0(0, Val, X1),      nth0(1, Val, Y1),      vache(X1, Y1, Race, vivante),      assert(vache(X1, Y1, Race, zombie)),      retract(vache(X1, Y1, Race, vivante)).  % le reste est le code prédéfini du jeu  initialisation :-      nombre\_rochers(NR),      placer\_rochers(NR),      nombre\_arbres(NA),      placer\_arbres(NA),      nombre\_vaches(brune, NVB),      placer\_vaches(brune, NVB),      nombre\_vaches(simmental, NVS),      placer\_vaches(simmental, NVS),      nombre\_vaches(alpine\_herens, NVH),      placer\_vaches(alpine\_herens, NVH),      placer\_dimitri,      creer\_zombie,      !.      affichage(L, \_) :-      largeur(L),      nl.      affichage(L, H) :-      rocher(L, H),      print('O'),      L\_ is L + 1,      affichage(L\_, H).      affichage(L, H) :-      arbre(L, H),      print('T'),      L\_ is L + 1,      affichage(L\_, H).      affichage(L, H) :-      dimitri(L, H),      print('D'),      L\_ is L + 1,      affichage(L\_, H).      affichage(L, H) :-      vache(L, H, brune, vivante),      print('B'),      L\_ is L + 1,      affichage(L\_, H).    affichage(L, H) :-      vache(L, H, brune, zombie),      print('b'),      L\_ is L + 1,      affichage(L\_, H).      affichage(L, H) :-      vache(L, H, simmental, vivante),      print('S'),      L\_ is L + 1,      affichage(L\_, H).    affichage(L, H) :-      vache(L, H, simmental, zombie),      print('s'),      L\_ is L + 1,      affichage(L\_, H).      affichage(L, H) :-      vache(L, H, alpine\_herens, vivante),      print('H'),      L\_ is L + 1,      affichage(L\_, H).    affichage(L, H) :-      vache(L, H, alpine\_herens, zombie),      print('h'),      L\_ is L + 1,      affichage(L\_, H).      affichage(L, H) :-      \+ occupe(L, H),      print('.'),      L\_ is L + 1,      affichage(L\_, H).      affichage(H) :-      hauteur(H).      affichage(H) :-      hauteur(HMax),      H < HMax,      affichage(0, H),      H\_ is H + 1,      affichage(H\_).      affichage :-      affichage(0),!.    %   jouer :-  %     initialisation,  %     tour(0, \_).  %   tour\_(\_, \_) :-  %     \+ verification,  %     write('Dimitri s\'est fait mordre'),!.  %   tour\_(N, R) :-  %     verification,  %     M is N + 1,  %     tour(M, R).  %   tour(N, R) :-  %     affichage,  %     question(R),  %     deplacement\_joueur(R),  %     deplacement\_vaches,  %     zombification,  %     tour\_(N, R). |

Affichage :

A picture containing text

Description automatically generated