A.

food(burger).

food(sandwich).

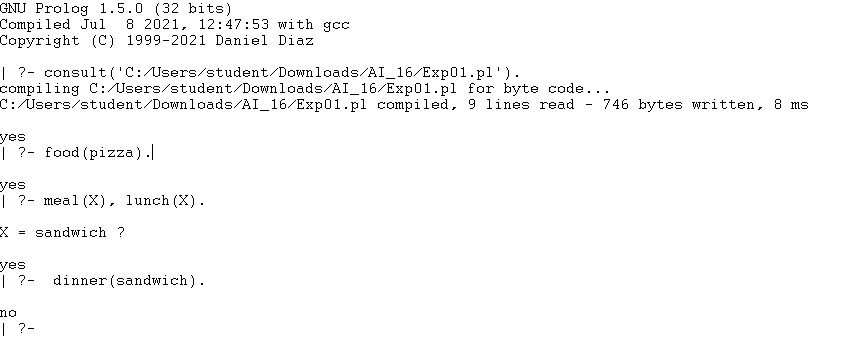
food(pizza).

lunch(sandwich).

dinner(pizza).

meal(X) :- food(X).

Output:



B.

studies(charlie, csc135).

studies(olivia, csc135).

studies(jack, csc131).

studies(arthur, csc134).

teaches(kirke, csc135).

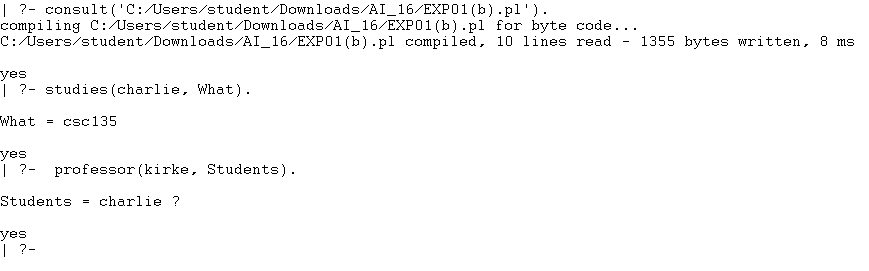
teaches(collins, csc131).

teaches(collins, csc171).

teaches(juniper, csc134).

professor(X, Y) :- teaches(X, C), studies(Y, C).

Output:



C.

% Jug capacities

capacity(a, 4).

capacity(b, 3).

% Initial state

initial(state(0, 0)).

% Goal: either jug has exactly 2 liters

goal(state(2, \_)).

goal(state(\_, 2)).

% Move: fill jug A

move(state(A, B), state(4, B)).

% Move: fill jug B

move(state(A, B), state(A, 3)).

% Move: empty jug A

move(state(A, B), state(0, B)) :- A > 0.

% Move: empty jug B

move(state(A, B), state(A, 0)) :- B > 0.

% Move: pour A -> B

move(state(A, B), state(NA, NB)) :-

A > 0,

capacity(b, CB),

SpaceB is CB - B,

Transfer is min(A, SpaceB),

NA is A - Transfer,

NB is B + Transfer.

% Move: pour B -> A

move(state(A, B), state(NA, NB)) :-

B > 0,

capacity(a, CA),

SpaceA is CA - A,

Transfer is min(B, SpaceA),

NA is A + Transfer,

NB is B - Transfer.

% Path search - avoid loops with visited states

solve(State, \_, []) :- goal(State).

solve(State, Visited, [Move|Moves]) :-

move(State, Move),

\+ member(Move, Visited),

solve(Move, [Move|Visited], Moves).

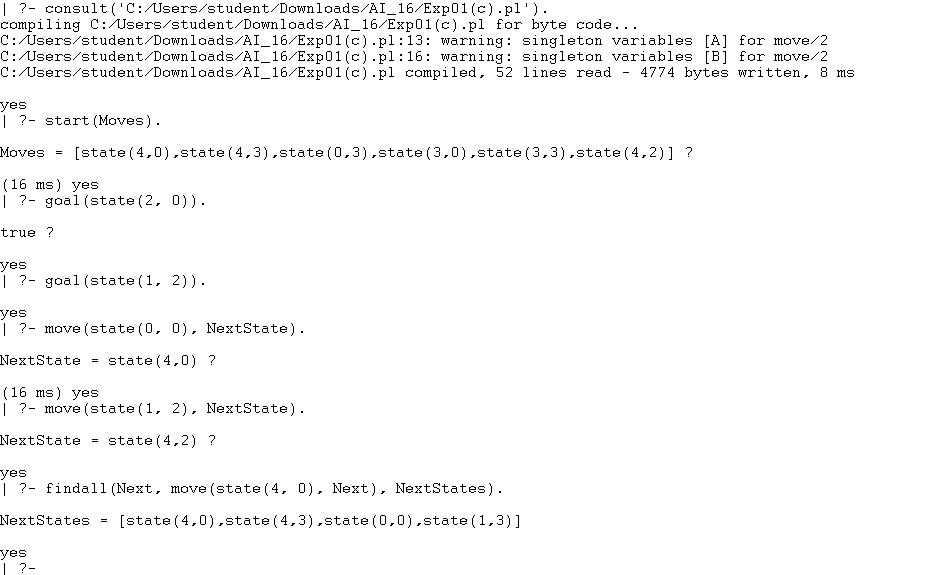
% Start solving from initial state

start(Moves) :-

initial(State),

solve(State, [State], Moves).

Output:



D.

assigned\_to(alex, alpha).

assigned\_to(julia, beta).

assigned\_to(sam, alpha).

assigned\_to(kate, gamma).

assigned\_to(ryan, gamma).

assigned\_to(nina, beta).

assigned\_to(liam, delta).

assigned\_to(emma, delta).

assigned\_to(zoe, alpha).

leads(john, alpha).

leads(susan, beta).

leads(john, gamma).

leads(michael, delta).

manager\_of(Manager, Employee) :- leads(Manager, Project), assigned\_to(Employee, Project).

teammates(E1, E2) :- assigned\_to(E1, Project), assigned\_to(E2, Project),E1 \= E2.

employees\_under(Manager, List) :-findall(E, manager\_of(Manager, E), List).

projects\_of(Manager, List) :-findall(P, leads(Manager, P), List).

team(Project, List) :- findall(E, assigned\_to(E, Project), List).

Output:

