**24.Write a Prolog program to implement Monkey**

**Banana Problem**

**Program:**

% Define the initial state and final state

initial\_state(state(at\_door, on\_floor, at\_window, has\_not\_eaten)). final\_state(state(\_, \_, \_, has\_eaten)).

% Define the possible actions and their effects

% Note: Action effects are expressed as changes to the state of the monkey

action(state(at\_door, on\_floor, at\_window, has\_not\_eaten), climb, state(at\_window, on\_window, at\_window, has\_not\_eaten)).

action(state(at\_window, on\_floor, at\_window, has\_not\_eaten), grasp, state(at\_window, on\_floor, at\_window, has\_eaten)).

action(state(at\_window, on\_window, at\_window, has\_not\_eaten), climb, state(at\_door, on\_window, at\_door, has\_not\_eaten)).

action(state(at\_door, on\_window, at\_door, has\_not\_eaten), walk, state(at\_middle, on\_floor, at\_middle, has\_not\_eaten)).

action(state(at\_middle, on\_floor, at\_middle, has\_not\_eaten), grasp, state(at\_middle, on\_floor, at\_middle, has\_eaten)).

action(state(at\_middle, on\_floor, at\_middle, has\_eaten), walk, state(at\_door, on\_floor, at\_door, has\_eaten)).

% Define a predicate to execute a sequence of actions

% Note: The last argument of the predicate is the final state of the sequence of actions execute\_actions(\_, [], FinalState) :- final\_state(FinalState). execute\_actions(CurrentState, [Action|Rest], FinalState) :- action(CurrentState, Action, NextState), execute\_actions(NextState, Rest, FinalState).

% Define a predicate to solve the problem

% Note: The solution is expressed as a sequence of actions solve\_problem(ActionList) :-

initial\_state(InitialState), execute\_actions(InitialState, ActionList, FinalState), final\_state(FinalState).

% Sample query and expected output

%

% Query: solve\_problem(ActionList).

% Expected output: ActionList = [climb, grasp, climb, walk, grasp, walk].

**Output:**

