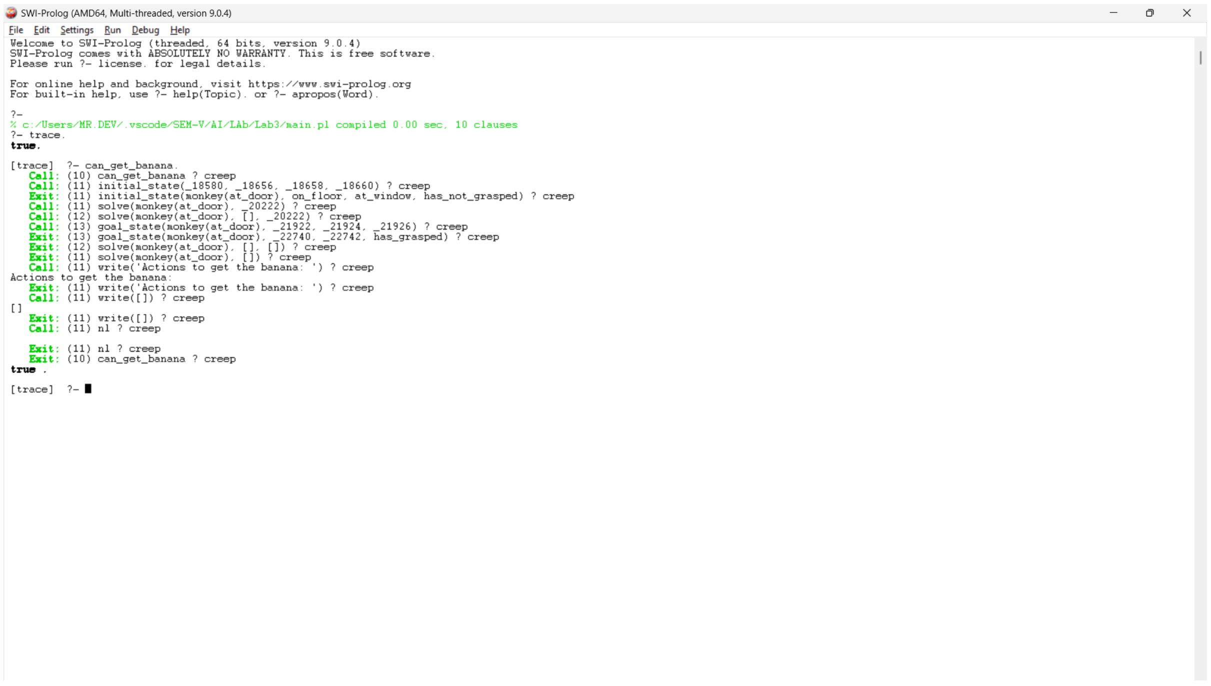
AI Lab3

> AP21110011577

> Chukka Chanakya Devendra

Question 1

```
% Define the initial state of the monkey, banana, and box
initial_state(monkey(at_door), on_floor, at_window, has_not_grasped).

% Define the state when the monkey is holding the banana
goal_state(_, _, _, has_grasped).

% Actions the monkey can perform
% The arguments are: Action, Pre-Monkey State, Pre-Banana State, Post-Monkey State, Post-Banana State

% Action: Walking on the floor
perform_action(walk, monkey(at_door), BananaState, monkey(in_room), BananaState).

% Action: Climbing the box
perform_action(climb, monkey(Location), BananaState, monkey(on_box), BananaState) :-
    Location \= on_box.

% Action: Pushing the box
perform_action(push, monkey(Location), BananaState, monkey(NewLocation), BananaState) :-
    Location \= at_window,
    NewLocation \= on_box.

% Action: Grasping the banana
perform_action(grasp, monkey(on_box), BananaState, monkey(on_box), BananaState) :-
    BananaState = BananaLocation,
    BananaLocation \= ceiling.

% Define the procedure to solve the problem
solve(State, Path) :-
    solve(State, [], Path).

solve(State, Path, Path) :-
    goal_state(State, _, _, _).

solve(State, Visited, Path) :-
    \+ member(State, Visited),  % Avoid revisiting states
    perform_action(_, State, _, NewState, _),
    solve(NewState, [State|Visited], Path).

% Query to check if the monkey can get the banana
can_get_banana :-
    initial_state(InitialMonkeyState, _, _, _),
    solve(InitialMonkeyState, Path),
    write('Actions to get the banana: '), write(Path), nl.
```


SWI-Prolog (AMD64, Multi-threaded, version 9.0.4)
File Edit Settings Run Debug Help

```
Welcome to SWI-Prolog (threaded, 64 bits, version 9.0.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?-
% c:/Users/MR.DEV/.vscode/SEM-V/AI/LAb/Lab3/main.pl compiled 0.00 sec, 10 clauses
?- trace.
true.

[trace]  ?- can_get_banana.
   Call: (10) can_get_banana ? creep
   Call: (11) initial_state(_18580, _18656, _18658, _18660) ? creep
   Exit: (11) initial_state(monkey(at_door), on_floor, at_window, has_not_grasped) ? creep
   Call: (11) solve(monkey(at_door), _20222) ? creep
   Call: (12) solve(monkey(at_door), [], _20222) ? creep
   Call: (13) goal_state(monkey(at_door), _21922, _21924, _21926) ? creep
   Exit: (13) goal_state(monkey(at_door), _22740, _22742, has_grasped) ? creep
   Exit: (12) solve(monkey(at_door), [], []) ? creep
   Exit: (11) solve(monkey(at_door), []) ? creep
   Call: (11) write('Actions to get the banana: ') ? creep
Actions to get the banana:
   Exit: (11) write('Actions to get the banana: ') ? creep
   Call: (11) write([]) ? creep
[]
   Exit: (11) write([]) ? creep
   Call: (11) nl ? creep

   Exit: (11) nl ? creep
   Exit: (10) can_get_banana ? creep
true .

[trace]  ?- █
```

Question 2

```
% Define a structured representation for rectangles
rectangle(width(Width), height(Height)).
```

In this representation, the functor `rectangle` takes two attributes: `width` and `height`, both of which are represented as functor terms themselves