PROGRAM 25

MONKEY BANANA PROBLEM

CODE:

```
% Initial state: monkey is at point (1, 1), banana is at point (4, 4).

% The grid is represented as (Row, Column).


% move(State1, Move, State2): Define possible moves and their effects.

move(state(Monkey, on_floor, Box, Has_not), grasp, state(Monkey, on_floor, Box, Has)) :-
    Monkey = (X, Y),
    Banana = (4, 4),
    Monkey \= Banana,
    Monkey \= Box.


move(state(Monkey, on_box, Box, Has_not), grasp, state(Monkey, on_box, Box, Has)) :-
    Monkey = Banana.


move(state(X, on_floor, X, Has_not), climb, state(X, on_box, X, Has_not)).
```

```
move(state(X, on_box, X, Has_not), push(X, Y), state(Y,
on_box, Y, Has_not)) :-
    not(Y = Banana),
    not(Y = (4,4)).

% Define the path from initial state to the goal state.
path(state(_, _, _, Has), _, []) :-
    Has = has.

path(State1, History, [Move|OtherMoves]) :-
    move(State1, Move, State2),
    not(member(State2, History)),
    path(State2, [State2|History], OtherMoves).

% Solve the problem and output the solution.
solve :-
    InitialState = state((1,1), on_floor, (1,1), has_not),
    path(InitialState, [InitialState], Moves),
    print_moves(Moves).

% Print the sequence of moves.
print_moves([]).
```

```
print_moves([Move|OtherMoves]) :-
    write(Move), nl,
    print_moves(OtherMoves).
```

OUTPUT: