```
setupRoverParams();
%pose_initial =
pose_final = [600.000; -200.000; deg2rad(45)];
delta_t = .01;

%Set up the simulation by initializing parameters
t = 0;  %seconds
pose = [100; 200; deg2rad(35)]; %pose vector [x, y, theta]; cm and radians are the units
control = [0,0]; %control vector [vl, vr]; units are cm per second for each wheel
[v, w] = kinematics_diff_drive(control, P);
wanted_Angle = pose_final(3);
state = [pose; v; w; t];  %initialization state
drawCar(state, P);
orig_angle = pose(3);

%TODO define a function which navigates to the final pose point given the
%algorithms found in the book and a pre-defined program like I implemented
%in the example function.

temp_pose = pose; % keeps previous values of pose for ease of translation and rotation
for t = 0.01:delta_t:10
    % I want 2 total rotations and 1 movement
    % I'll allocate .2t to the rotations respectively and .6t to the
    % translation. That is to say, in a time scale of 10 seconds, I want to
    % spend 2 seconds rotating, then 6 seconds translating, then a final 2
    % seconds rotating to the final resting rotation.
    control(1) = 0;
    control(2) = 0;
    if(pose(1) ~= pose_final(1) || pose(2) ~= pose_final(2) || pose(3) ~= pose_final(3))%↙
check if in position and rotation
        if(~(pose(1)-.0001 < pose_final(1) && pose(1)+.0001 > pose_final(1)) && ~(pose(2)↙
-.0001 < pose_final(2) && pose(2)+.0001 > pose_final(2)))  % check if in position
            if(pose(3) ~= wanted_Angle) % not in position, make sure we're heading to↙
goal
                if(pose(3)-.0001 < wanted_Angle && pose(3) + .0001 > wanted_Angle)
                    pose(3) = wanted_Angle;
                else
                    hyp = sqrt((pose_final(1) - pose(1))^2 + (pose_final(2) - pose(2))↙
^2); % get hypotenuse
                    wanted_Angle = acos((pose_final(1) - pose(1))/hyp); % get heading↙
angle
                    if(pose_final(2) < 0)
                        wanted_Angle = -wanted_Angle;
                    end
                    control(1) = ((temp_pose(3) - wanted_Angle)/2)*(21.5)/2; % get the↙
speed of the right wheel for rotation
                    control(2) = -control(1); % get the speed of the left wheel for↙
rotation
                end
            else % heading achieved, move towards goal
                temp_pose(3) = pose(3); % store current heading for use in rotating from↙
```

```
heading to final
                hyp = sqrt((pose_final(1) - temp_pose(1))^2 + (pose_final(2) - temp_pose↙
(2))^2); % get hypotenuse
                control(1) = hyp/6;
                control(2) = control(1);
            end
        elseif (pose(3) ~= pose_final(3)) % We're in position, but our rotation isn't↙
done
            if(pose(3)-.0001 < pose_final(3) && pose(3) + .0001 > pose_final(3))
                    pose(3) = pose_final(3);
            else
                control(1) = ((temp_pose(3) - pose_final(3))/2)*(21.5)/2; % get the speed↙
of the right wheel for rotation
                control(2) = -control(1); % get the speed of the left wheel for rotation
            end
        end
    end
    [v, w] = kinematics_diff_drive(control, P);
    pose = propagatePose(pose, v, w, delta_t);
    state = [pose; v; w; t];
    drawCar(state, P);
    %pause(.01);
end
```