1. Mobile Robot Path Planning and Control Simulation. The lower MATLAB code has uncompleted lines on purpose (lines shown in red). This code uses potential field approach to navigate a mobile robot (three-wheeled system) to avoid obstacles and reach a specific goal location. The intention is for you to complete the code (and perhaps to change part of it). The mobile robot starting position is (0,0), and final target is position (10,10), having obstacles at coordinate positions (3,0), and (7,9).

function xdot=NewPathMob(tspan,X);

%%% COPY FROM HERE
%Obstacles:
X1 = 3; Y1 = 0;
X2 = 7; Y2 = 9;

%Goal:
GoalX= 10; GoalY=10;

T=linspace(0,25,1000);
[t,X]=ode23('NewPathMob',T,[0 0 -pi/2 0]); %pointing towards one of the obstacles
figure(1), plot(X(:,1),X(:,2))

figure(2), plot(X(:,3)*180/pi), title('Phi')
figure(3), plot(X(:,4)*180/pi), title('Alpha')

%%% TO HERE to MATLAB Window !!!

% Specification of system (THREE-WHEELED MOBILE ROBOT)
X = [X(0)= x;  X(1) = y; X(3) = Phi; X(4) = alpha ];
L = 0.3;  % 30 cms.

k1 = ___; k2 = ___; k3 = ___;  % GAINS NOT SHOWN ON PURPOSE!
kp = ___;  % GAIN NOT SHOWN ON PURPOSE!
kv = ___;  % Error coefficient for velocity NOT SHOWN ON PURPOSE!

% Calculating the Potential vectors:
% Obstacle1 located at (3,0)
Mag1x = (X(1)-X1);
Mag1y = (X(2)-Y1);
if M1==0 M1=0.00001; end %due to the division below

% Obstacle2 located at (7,10)
Mag2x = (X(1)-X2);
Mag2y = (X(2)-Y2);
if M2==0 M2=0.00001; end %due to the division below

% Goal located at (10,10)
Mag3x = (X(1)-GoalX);
Mag3y = (X(2)-GoalY);
M3 = sqrt(Mag3x^2+Mag3y^2);
if M3==0 M3=0.00001; end %due to the division below

Fx = k1*Mag1x/M1 + k2*Mag2x/M2 - k3*Mag3x/M3;
Fy = k1*Mag1y/M1 + k2*Mag2y/M2 - k3*Mag3y/M3;

PhiD = atan2(-Fx,Fy);
% The distance of the desired curvature vector is the magnitude of the reaction force
R = sqrt(Fx^2+Fy^2);

% Velocity of the Wheel depends on the magnitude of the reaction/force
Vt = R/kp;

% Tracking Error in Phi (the direction of Phi is negative)
Perror = kp * PhiD-X(3);
if (Perror>pi/2)
Perror = Perror-pi/4;
end
if (Perror<-pi/2)
Perror = Perror+pi/4;
end

% Calculating Control angle Alpha:
alpha = ___ + atan2(L,R);  % Alpha depends on Perror
if (alpha>pi/2)
alpha = alpha-pi/4;
end
if (alpha<-pi/2)
alpha = alpha+pi/4;
end

xdot=[
Vt*(-sin(X(3)))*cos(alpha);  % Driven by desired alpha and Vt velocity
Vt*cos(X(3))*cos(alpha);  % Driven by desired alpha and Vt velocity
Vt*sin(alpha)/L;  % Controlled by desired alpha and Vt velocity
(alpha-X(4))*25/1000;  % Due that each simulation runs for 25/1000 seconds
];
For some specific gains used by the instructor, he got the following graphs. Try improving the performance of his results by using appropriate gains and maybe changing the control strategy shown in the code.

2. Simulation. Below is a Simulink MATLAB diagram showing the internal dynamics of a three-wheel mobile robotic system (it uses desired (x,y) path and it calculates needed alpha and velocity to maintain such desired trajectory.) This diagram uses the equations provided from Ollero’s book.

Write down a Simulink diagram to simulate the internal dynamics of a Differential configuration mobile robot system, the inputs should be the driving rotational velocities of Left and Right wheels, $w_L$ and $w_R$, and the output should be the final position (x,y) (i.e. plot X versus Y), and orientation angle of vehicle, $\phi$ (i.e. plot $\phi$ versus time). The system variables are defined as follows: $r=0.25$ m, $b=1$ m, the initial position is (0,0), and it initial orientation $\phi=-\pi/4$ rad. The simulation should run for 30 seconds for the following inputs ($w_L$, and $w_R$ pairs shown in next page):
a) $w_L$, and $w_R$ have same angular velocity (1 rad/sec).
b) $w_R$ has a little higher velocity (1.2 rad/sec) than $w_L$ (1 rad/sec).
c) $w_L$ has constant velocity (1 rad/sec), and $w_R$ starts from zero velocity but increasing by 0.1 rad/sec.
d) $w_L$, and $w_R$ have same angular velocity but opposite signs! (1 rad/sec, and -1 rad/sec respectively).