```
%hw 1
%% Author: Tatiana D. Luna
%%%%%%%%%%%%%%%%%%%%%%PART A%%%%%%%%%%%%%%%%%%%%%%
r0 = [8000; 0; 6000]; %km
v0 = [0; 8.5; 0]; %km/s

G = 6.6742*10^(-20); %km^3/(kg*s^2) gravitational constant
m1 = 5.972*10^24; %kg mass of Earth
m2 = 0; %negligible compared to Earth
mu = G*(m1+m2); %km^3/s^2 gravitational parameter

%1. computing orbital angular momentum
%h=rxv
hvector = cross(r0, v0)
h = norm(hvector)

%2.  eccentricity vector
evector = cross(v0, hvector)/mu - r0/norm(r0)
e = norm(evector);
fprintf('e = %f therefore, it is an elliptic\n',e);

%3. Distance from Earth's center at perigee.
rp = (h^2/mu)*(1/(1+e))
R = 6371; %Earth's radius in km
r_centerto_p = R+rp;
fprintf('The distance from Earths center at perigee is: %f Km
\n',r_centerto_p);


%4. Distance from Earth's center at apogee
ra = (h^2/mu)*(1/(1-e))
r_centerto_a = R+ra;
fprintf('The distance from Earths center at apogee is: %f Km
\n',r_centerto_a);

%5. Semi major axis (a)
a = (h^2/mu)*(1/(1-(e^2)));
fprintf('The semi major axis is: %f Km \n',a);

%6. Orbital period.
T = (2*pi/sqrt(mu))*a^(3/2);
fprintf('The orbial period is: %f s \n',T);


%%%%%%%%%%%%%%%%%%%%%%%%PART B%%%%%%%%%%%%%%%%%%5

%Use Matlab's ODE45 to integrate the equations of motion for 1 orbital
period,
%with a maximum time step of 10 seconds (OPTIONS = odeset('Maxstep', 10)).
%Plot the trajectory in ECI, using a red line, of width 2.

%need to find unit vector of moving frame
TSPAN = [0 T]; %from 0 to period
OPTIONS = odeset('Maxstep', 10);
```

```
Y0 = [r0; v0]; %initial state vector (km & km/s) aka initial conditions

%ode45 integrates from tspan (o to t) using initial conditions YO
%returns two columns tout and yout represents f(t,y), t being the period
[TOUT,YOUT] = ode45(@relacc,TSPAN,Y0,OPTIONS);

%defining the coordinate system fixed with orbital plane (perifocal
%coordinate system)
z_hat = hvector/h; %unit vector normal to orbital plane
x_hat = evector/e; %unit vector along e
y_hat = cross(z_hat, x_hat);

%plotting orbit from ODE45
figure(1)
plot3(YOUT(:,1),YOUT(:,2),YOUT(:,3),'r','linewidth',2)
grid on
hold on
plot3(YOUT(1,1),YOUT(1,2),YOUT(1,3),'r*')
xlabel('x (km)'); ylabel('y (km)'); zlabel('z (km)');
[X,Y,Z] = sphere;
surf(6378*X, 6378*Y, 6378*Z)


%%%%%%%%%%%%%%%%PART C%%%%%%%%%%%%%%%%%

%defining theta values
n = 360; %number of points per orbit (fineness of plot)
theta = linspace(0, 2*pi, n);
%for elliptical orbit:
r = (h^2/mu)*(1./(1+e*cos(theta)));
for i=1:n
    r2(:,i) = r(i)*cos(theta(i))*x_hat + r(i)*sin(theta(i))*y_hat;
end
%plotting from Keplerain orbit equation
plot3(r2(1,:),r2(2,:),r2(3,:),'black');


%%%%%%%%%%%%%%%%PART D%%%%%%%%%%%%%%%%%%%%%%%%%

%Compute the velocity the spacecraft should have at perigee to be in a
%circular orbit. Plot that circular orbit with a blue line of width 2,
%on the same figure

center=[0;0;0];
%r at perigee will be radius of circle, r=h^2/mu (for circle)
hcircle = sqrt(rp*mu);

%velocity of a circular orbit is mu/h and vr=0
v_tan = mu/hcircle;
fprintf('Velocity at perigee: %f km/s \n',v_tan);

%plottin the circular orbit
p=r2(:,1)'; %vector at perigee
perp_p=null(p); %%vector perpindicular to p
```

```
circle_3D(rp,center,perp_p(:,2),1);



%%%%%%%%%%%%%%%%%%%%%%%%PART E%%%%%%%%%%%%%%%%%%%%%%%%

%Compute the velocity the spacecraft should have at apogee to be in a
%circular orbit. Plot that circular orbit with a green line of width 2

hcircle2 = sqrt(ra*mu);

%velocity of a circular orbit is mu/h and vr=0
v_tan2 = mu/hcircle2;
fprintf('Velocity at apogee: %f km/s \n',v_tan2);

av = r2(:,181)'; %vector at apogee
perp_a=null(av); %vector perpindicular to av
circle_3D(ra,center,perp_a(:,2),2);
axis equal


%without changing its position, what are the velocity corrections (?V)
%required to circularize the initial orbit at perigee and at apogee?

%velocity at perigee is 6.31 km/s when in circular orbit
deltaV_perigee = v0(2,1)-v_tan;
fprintf('Velocity correction at perigee: %f km/s \n',deltaV_perigee);
%deltaV_apogee = v0(2,1)- v_tan2; v0 is the velocity at perigee thus wrong
deltaV_apogee =-(((mu/h)*(1-e))-v_tan2);
fprintf('Velocity correction at apogee: %f km/s \n',deltaV_apogee);
```