```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This code is applicable to both problems 1 and 2 of Homework 4
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [T,U] = RKw17sc(odefun,TSPAN,U0,TOL,A,b,c,qorder,mu);
% RK method with stepsize control using Richardson extrapolation
% qorder = order of method used

dtmin = 1.e-5; % minimum allowed step size
sf = 0.8; %safety factor

q1 = qorder+1;
T0 = TSPAN(1); TFINAL = TSPAN(2);
dt = (TFINAL-T0)/10;
m = length(U0);
U = U0;
T(1)=T0;
kstep = 1;
while T(kstep) < TFINAL
    t = T(kstep);
    flag = 1;
    while flag == 1;
        %one big step with dt
        Ubig = RKexplicitstep(odefun,t,U(:,kstep),dt,A,b,c,mu);

        % two small steps with dt/2
        dt2 = dt/2;
        V = RKexplicitstep(odefun,t,U(:,kstep),dt2,A,b,c,mu);
        Usmall = RKexplicitstep(odefun,t+dt2,V,dt2,A,b,c,mu);

        %Richardson extrapolation
        fac = 2^qorder;
        Unew = (fac*Usmall-Ubig)/(fac-1);

        %Estimate of one-step error for Usmall method
        locerr =norm(Unew-Usmall,1);

         % conditions for accepting current dt
        if (locerr <= TOL ) | (dt < 1.001*dtmin)
            flag = 0;
            kstep = kstep+1;
            U(:,kstep) = Unew; %local extrapolation
            T(kstep) = t + dt;
        end
        dt = max(sf*((TOL/locerr)^(1/q1))*dt,dtmin); %new value for dt
        dt = min(dt,TFINAL-t);
    end
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function U = eulerstep(odefun,t,U0,dt,mu)
% Takes one step of the foward Euler method
global steps;
U = U0 + dt*feval(odefun,t,U0,mu);
steps = steps + 1;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function yprime = orbitODE(t,y,mu)
% Implements the ODE from Homework 2 Problem 3 as a first order system
muhat = 1.0 - mu;
D1 = ((y(1) + mu)^2 + y(2)^2)^1.5;
D2 = ((y(1) - muhat)^2 + y(2)^2)^1.5;
yprime = zeros(size(y));
yprime(1) = y(3);
yprime(2) = y(4);
yprime(3) = y(1) + 2*y(4) - muhat*((y(1)+mu)/D1) - mu*((y(1)-muhat)/
D2);
yprime(4) = y(2) - 2*y(3) - (muhat*(y(2)/D1)) - (mu*(y(2)/D2));
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function U = RKexplicitstep(odefun,t,U0,dt,A,b,c,mu)
% One step of a general explicit Runge Kutta method
% A is assumed to be strictly lower triangular
% b and c must be column vectors
%
global steps;
r = length(b);
m = length(U0);
K = zeros(m,r);%matrix whose j-th column is K_j
K(:,1) = feval(odefun,t,U0,mu);
for j = 2:r
    Y = U0 + dt*K(:,1:j-1)*(A(j,1:j-1).');
     % Y = U0 + dt*sum_{l=1}^{j-1} a_{jl}K_l
    K(:,j) = feval(odefun,t + c(j)*dt, Y,mu);
end
U = U0 + dt*K*b;
steps = steps + 1;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function yprime = hw4p2(t,u,mu)
% ODE specific to problem 2 of homework 4
if (round(t,12) == 0 & round(u,12) == 0)
    yprime = 0;
else
    yprime = (2*t*u)/(t^2 + u^2);
end
end
```