```
function [path, distance, vel_out, g_s] = banked_turn(vel_in, pos_in,
 n, r, turn_dir, bank_angle, h0, path_ref)


    % Running sequence as normal
    [path, distance, vel_out] = bankedTurn_compute_path(pos_in,
 vel_in, n, turn_dir, r, h0);

    if isnan(path_ref)
        % Running sequence as normal
        g_s = bankedTurn_compute_g_s(path, h0, r, turn_dir,
 bank_angle);
    else
        % Using dependent input as reference... used to bypass
 unimplemented functions
        g_s = bankedTurn_compute_g_s(path, h0, r, turn_dir,
 bank_angle);
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% COMPLETE FUNCTIONS BELOW %%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%

function [path, distance, vel_out] = bankedTurn_compute_path(pos_in,
 vel_in, n, turn_dir, r, h0)

     %create theta vector for the turn and the xyz coordinates.  Make
 sure the
    %loop starts at the beginning x, y, and z coordinates
    g = -9.81;

    xin = pos_in(1);
    yin = pos_in(2);
    zin = pos_in(3);

    % starting velocities
    vx = vel_in(1);
    vy = vel_in(2);

    % starting angle of loop
    theta_in = sign(vy)*pi/2 + sign(vx)*(pi/2+sign(vx)*pi/2);
    theta_vec = linspace(theta_in, theta_in-pi*sign(turn_dir), n);

    % converting theta to xy position - split into cw and ccw case
    if turn_dir == -1 %ccw
        if sign(vy) > 0
            x = r*sin(theta_vec) - r*sin(theta_in) + xin;
            y = r*cos(theta_vec) - r*cos(theta_in) + yin;
        else
            x = r*sin(theta_vec - pi) - r*sin(theta_in) + xin;
            y = r*cos(theta_vec - pi) - r*cos(theta_in) + yin;
        end
```

```
    else %cw
        if sign(vy) > 0
            x = -r*sin(theta_vec) + r*sin(theta_in) + xin;
            y = -r*cos(theta_vec) +  r*cos(theta_in) + yin;
        else
            x = -r*sin(theta_vec - pi) + r*sin(theta_in) + xin;
            y = -r*cos(theta_vec - pi) +  r*cos(theta_in) + yin;
        end
    end

    path = [x', y', ones(n,1)*zin];

    % computed at each point along loop
    distance = r*(theta_vec'-theta_vec(1))*-sign(turn_dir);

    %[vx vy vz] at loop exit
    mag = sqrt(2*g*(zin-h0));

    if sign(vy) > 0
        vel_out = [mag*cos(theta_vec(end)),-
mag*sin(theta_vec(end)),0];
    else
        vel_out = [mag*cos(theta_vec(end)),mag*sin(theta_vec(end)),0];
    end
    vel_out = [mag*cos(theta_vec(end)),mag*sin(theta_vec(end)),0];
end

function g_s = bankedTurn_compute_g_s(path, h0, r, turn_dir,
 bank_angle)
    % Number of elements in path matrix
    z = path(:,3);
    n = length(z);

    g = 9.81;

    % Pos ccw, neg cw

    % hint: solve system of equations for F_normal and F_lateral
    v = sqrt(2*g*(h0-z));

    N = (g*r*cos(bank_angle) + (v.^2)*sin(bank_angle))/r;

    L = (g*r*sin(bank_angle) - (v.^2)*cos(bank_angle))/r;

    %Compile the g forces and xyz coordinates into the matrices to be
 outputted
    %Gs matrix [front/back, left/right, up/down]... Front, left and up
 are defined as position
    % cos(bank_angle) + ((v.^2)*sin(bank_angle))/(r*g)
    % (sin(bank_angle) - ((v.^2)*cos(bank_angle))/
(r*g))*sign(turn_dir)
    g_s = [zeros(n,1), (sin(bank_angle) - ((v.^2)*cos(bank_angle))/
(r*g))*sign(turn_dir), cos(bank_angle) + ((v.^2)*sin(bank_angle))/
(r*g)];
```

```
end
```

```
Not enough input arguments.

Error in banked_turn (line 5)
    [path, distance, vel_out] = bankedTurn_compute_path(pos_in,
 vel_in, n, turn_dir, r, h0);
```

Published with MATLAB® R2021a