Problem 1

```
disp('');

% a) AirRelativeVelocityToWindAngles
function [wind_angles] = AirRelativeVelocityVectorToWindAngles(velocity_body)
% Calculates the wind angles and air speed given the air relative velocity
% vector
% Inputs:
%   velocity_body -> The air relative velocity vector in body coordinates
% Output:
%   wind_angles -> [air speed; sideslip angle; angle of attack]
%
% Author: Thomas Dunnington
% Date Modified: 8/30/2024

Va = norm(velocity_body);
beta = asin(velocity_body(2) / Va);
alpha = atan2(velocity_body(3), velocity_body(1));

wind_angles = [Va; beta; alpha];
end


% b) WindAnglesToAirRelativeVelocityVector
function [velocity_body] = WindAnglesToAirRelativeVelocityVector(wind_angles)
% Calculates the air relative velocity vector given the wind angles and air
% relative speed
% Inputs:
%   wind_angles -> [air speed; sideslip angle; angle of attack]
% Output:
%   velocity_body -> The air relative velocity vector in body coordinates
%
% Author: Thomas Dunnington
% Date Modified: 8/30/2024

Va = wind_angles(1);
beta = wind_angles(2);
alpha = wind_angles(3);

velocity_body = Va .* [cos(alpha)*cos(beta); sin(beta);
sin(alpha)*cos(beta)];
end


% c) RotationMatrix321
function R = RotationMatrix321(euler_angles)
% Calculates the 3-2-1 rotation matrix given a set of yaw, pitch, and roll
% Euler angles
% Inputs:
%   euler_angles -> [roll; pitch; yaw]
% Output:
```

```
%   R -> 3x3 Transformation matrix
%
% Author: Thomas Dunnington
% Date Modified: 8/30/2024

phi = euler_angles(1);
theta = euler_angles(2);
psi = euler_angles(3);

R = [cos(theta)*cos(psi), cos(theta)*sin(psi), -sin(theta);
    sin(phi)*sin(theta)*cos(psi) - cos(phi)*sin(psi),
sin(phi)*sin(theta)*sin(psi)+cos(phi)*cos(psi), sin(phi)*cos(theta);
    cos(phi)*sin(theta)*cos(psi)+sin(phi)*sin(psi),
cos(phi)*sin(theta)*sin(psi)-sin(phi)*cos(psi), cos(phi)*cos(theta)];
end


% d) TransformFromBodyToInertial
function vector_inertial = TransformFromBodyToInertial(vector_body,
euler_angles)
% Converts the body coordinates vector to inertial coordinates using the
provided euler
% angles
% Inputs:
%   euler_angles -> [roll; pitch; yaw]
%   vector_body -> [x_B; y_B; z_B] arbitrary vector in body coordinates
% Output:
%   vector_inertial -> [x_E; y_E, z_E] vector in inertial coordinates
%
% Author: Thomas Dunnington
% Date Modified: 8/30/2024

vector_inertial = RotationMatrix321(euler_angles)' * vector_body;
end


% e) TransformFromInertialToBody
function vector_body = TransformFromInertialToBody(vector_inertial,
euler_angles)
% Converts the inertial vector to body coordinates using the provided euler
% angles
% Inputs:
%   euler_angles -> [roll; pitch; yaw]
%   vector_inertial -> [x_E; y_E, z_E] arbitrary inertial vector
% Output:
%   vector_body -> [x_B; y_B; z_B] vector in body coordinates
%
% Author: Thomas Dunnington
% Date Modified: 8/30/2024

vector_body = RotationMatrix321(euler_angles)*vector_inertial;
end
```

Published with MATLAB® R2023b