```
%==========================
%  Name:                hw3_1.m
%
%  Author:              Kairi Kozuma
%
%==========================

% Transformation matrices
T_WC = [-10;15;-5];
R_WC = [0.836516303737808,0.224143868042013,0.500000000000000;0.0173375885302536,0.901221065013438,-0.433012701892219;-0.547667674420164,0.37089097912

% a) Find camera frame points from world frame points

    % Points in world frame
    p1_W = [6.60000000000000,-7.20000000000000,-41.1000000000000,-13.5000000000000;-1.70000000000000,10.5000000000000,10.7000000000000,18.300000000000

    % Convert to camera frame
    qc1 = transformToCamera(p1_W, R_WC', -R_WC'*T_WC);

    % Print out values
    fprintf('a) Points in camera frame:\n');
    disp (qc1(1:3,:));

% b) Determine which points are in field of view

    % Field of view +- the following value
    horiFOV = 45;
    vertFOV = 30;

    % Determine if in field of view
    inView = inFOV(qc1(1:3,:),horiFOV, vertFOV);

    fprintf('b) Points in field of view:\n');
    count = 0;
    for n = 1:length(inView)
        if (inView(n))
            fprintf('Point q%d in field of view\n', n);
            count = count + 1;
        end
    end

    if (count == 0)
        fprintf('\tNo points in field of view\n\n');
    end


% c) Find world frame points from camera frame points

    % Points in camera frame
    p2_C = [2.80000000000000,13.1000000000000;1.40000000000000,-11.3000000000000;16,28];

    % Conver to world frame
    qw2 = transformToWorld(p2_C, R_WC, T_WC);

    % Print out values
    fprintf('c) Points in world frame:\n');
    disp (qw2(1:3,:));

%======================== transformToCamera ================================
%
%  qc = transformToCamera(pw, R_CW, T_CW)
%
%  INPUTS:
%    pw         - point in 3 dimension, world frame
%    R_CW       - rotation matrix
%    T_CW       - translation vector
%
%  OUTPUTS:
%    qc             - point in 3 dimensions, camera frame
%
%======================== transformToCamera ================================
function [pc] = transformToCamera(pw, R_CW, T_CW)

transformMatrix = [R_CW,T_CW;0,0,0,1];

dim = size(pw);
lastRow = ones([1,dim(2)]);

qw = [pw; lastRow];

pc = transformMatrix * qw;

end

%=================================== inFOV ================================
%
%  inView = inFOV(pc, horiFOV, vertFOV)
%
%  INPUTS:
%    pc         - point in 3 dimensions, camera frame
```

```
%    horiFOV    - horizontal field of view, +- value
%    vertFOV    - vertical field of view, +- value
%
%  OUTPUTS:
%    inView             - boolean vector of whether points are in FOV
%
%================================== inFOV ================================
function [inView] = inFOV(pc, horiFOV, vertFOV)

angleY = (180 / pi) * atan2(pc(2,:), pc(3,:));
angleX = (180 / pi) * atan2(pc(1,:), pc(3,:));

angles = [angleX; angleY];

inHoriView = (angles(1,:) >= -horiFOV & angles(1,:) <= horiFOV);
inVertView = (angles(2,:) >= -vertFOV & angles(2,:) <= vertFOV);

inView = inHoriView & inVertView;

end

%======================== transformToWorld ================================
%
%  qw = transformToWorld(pc, R_WC, T_WC)
%
%  INPUTS:
%    pc         - point in 3 dimensions, camera frame
%    R_WC       - rotation matrix
%    T_WC       - translation vector
%
%  OUTPUTS:
%    qw             - point in 3 dimensions, world frame
%
%======================== transformToWorld ================================
function [pw] = transformToWorld(pc, R_WC, T_WC)

transformMatrix = [R_WC,T_WC;0,0,0,1];

dim = size(pc);
lastRow = ones([1,dim(2)]);

qc = [pc; lastRow];

pw = transformMatrix * qc;

end
```

```
a) Points in camera frame:
    6.6960   -8.4153  -21.4898   -1.5562
   -6.6564    3.8045  -13.9616    1.2994
   24.9813   17.9736  -19.9880   -4.9789

b) Points in field of view:
Point q1 in field of view
Point q2 in field of view
c) Points in world frame:
    0.6560   12.4255
    9.3821   -7.0810
    5.9858    4.6345
```

Published with MATLAB® R2016b