```
%% Question 2

function x_star = xStar(m)
    x_star = zeros(m, 1);
    for i = 1:m
        x_star(i) = (-1)^i * (i/(i+1));
    end
end

% 2(a)
H_5 = hilb(5);
H_10 = hilb(10);
H_20 = hilb(20);
[Q_5, R_5] = qr(H_5);
[Q_10, R_10] = qr(H_10);
[Q_20, R_20] = qr(H_20);

% 2(b)

% Below are functions from Homework 1.
% Since we don't have control over the "\" operator,
% we'll take the naiive approach to illustrate our point.

function U = myGaussianElimination(A)
    [m, n] = size(A);
    if n ~= m + 1
        error("Expecting an n x (n+1) matrix, got: %d x %d", m, n)
    end

    for i = 1:m
        if A(i,i) == 0
            error("Zero pivot at row %d", i);
        end
        for j = i+1:m  % For each row entry below the pivot
            ell_ij = A(j,i) / A(i,i);
            A(j,:) = A(j,:) - ell_ij * A(i,:);  % Apply row-op
        end
    end

    U = A;
end

function x = myBackwardSubstitution(U, c)
    [m, n] = size(U);
    if n ~= m
        error("`U` must be an n x n matrix, got: %d x %d", m, n)
    end

    [p, q] = size(c);
    if q ~= 1
        error("`c` must be a column vector (1 x m), got: %d x %d", p, q)
    elseif p ~= m
        error("Expecting column vector with %d rows, got: %d", m, p)
    end

    x = zeros(m, 1);
    for i = m:-1:1
        if U(i,i) == 0
            error("Zero diagonal entry at row %d", i);
        end
        sum = 0;
        for j = i+1:n  % Perform summation, idk how to with one-line??
            sum = sum + U(i,j) * x(j);
        end
        x(i) = (1 / U(i,i)) * (c(i) - sum);
    end
end

function x = myLinearSolution(A, b)
    [m, n] = size(A);
    if m ~= n
        error("`A` must be an n x n matrix, got: %d x %d", m, n)
    end

    [p, q] = size(b);
    if q ~= 1
        error("`b` must be a column vector (1 x m), got: %d x %d", p, q)
    elseif p ~= m
        error("Expecting column vector with %d rows, got: %d", m, p)
    end

    augmented = [A b];
    echelon = myGaussianElimination(augmented);
    U = echelon(:, 1:m);
    c = echelon(:, n+1);
    x = myBackwardSubstitution(U, c);
end

b_star_5 = H_5 * xStar(5);
b_star_10 = H_10 * xStar(10);
b_star_20 = H_20 * xStar(20);

% Solve using Gaussian elimination.
x_gauss_5 = myLinearSolution(H_5, b_star_5)
x_gauss_10 = myLinearSolution(H_10, b_star_10)
x_gauss_20 = myLinearSolution(H_20, b_star_20)

% Solve using QR factorizations found in 2(a).
% For simplicity here, we'll use the "\" operator to solve.
x_qr_5 = R_5 \ (Q_5' * b_star_5)
x_qr_10 = R_10 \ (Q_10' * b_star_10)
x_qr_20 = R_20 \ (Q_20' * b_star_20)

% 2(c)
x_delta_gauss_5 = norm(x_gauss_5 - xStar(5))
x_delta_gauss_10 = norm(x_gauss_10 - xStar(10))
x_delta_gauss_20 = norm(x_gauss_20 - xStar(20))

x_delta_qr_5 = norm(x_qr_5 - xStar(5))
x_delta_qr_10 = norm(x_qr_10 - xStar(10))
x_delta_qr_20 = norm(x_qr_20 - xStar(20))

%% Question 3

function H = myHouseholder(v, w)
    [len_v, n] = size(v);
    if n ~= 1
        error("Expecting a column vector `v`, got: %d x %d", m, len_v)
    end
    [len_w, n] = size(w);
    if n ~= 1
        error("Expecting a column vector `w`, got: %d x %d", m, len_w)
    end
    if len_v ~= len_w
        error("Mismatch dimensions, `v` is in R^%d but `w` is in R^%d", len_v,
len_w)
    end

    % Normalize.
    v_hat = v / norm(v);
    w_hat = w / norm(w);

    % Edge case: if they are the same vectors
    % Or divide by zero will occur!!!
    if v_hat == w_hat
        H = eye(len_w);
        return
    end

    % Apply Householder expression.
    u = v_hat - w_hat;
    H = eye(len_w) - 2 * (u * u') / (norm(u)^2);
end

% Assuming the question means three PAIRS of vectors.
disp('Test 1:');
v_1 = randn(4, 1);
w_1 = randn(4, 1);
H_1 = myHouseholder(v_1, w_1);
disp(H_1 * (v_1 / norm(v_1)));
disp(w_1 / norm(w_1));

disp('Test 2:');
v_2 = randn(4, 1);
w_2 = randn(4, 1);
H_2 = myHouseholder(v_2, w_2);
disp(H_2 * (v_2 / norm(v_2)));
disp(w_2 / norm(w_2));

disp('Test 3:');
v_3 = randn(4, 1);
w_3 = randn(4, 1);
H_3 = myHouseholder(v_3, w_3);
disp(H_3 * (v_3 / norm(v_3)));
disp(w_3 / norm(w_3));
```