```
function [s, mu0] = solveqp(x, W, df, g, dg)
    % Implement an Active-Set strategy to solve the QP problem given by
    % min     (1/2)*s'*W*s + c'*s
    % s.t.    A*s-b <= 0
    %
    % where As-b is the linearized active contraint set

    % Strategy should be as follows:
    % 1-) Start with empty working-set
    % 2-) Solve the problem using the working-set
    % 3-) Check the constraints and Lagrange multipliers
    % 4-) If all constraints are staisfied and Lagrange multipliers are positive,
terminate!
    % 5-) If some Lagrange multipliers are negative or zero, find the most negative one
    %     and remove it from the active set
    % 6-) If some constraints are violated, add the most violated one to the working
set
    % 7-) Go to step 2


    % Compute c in the QP problem formulation
     c = df(x)';

    % Compute A in the QP problem formulation using all constraints
     A0 = dg(x);

    % Compute b in the QP problem formulation using all constraints
     b0 = -1*g(x);

    % Initialize variables for active-set strategy
    stop = 0;           % Start with stop = 0
    % Start with empty working-set
    A = [];         % A for empty working-set
    b = [];         % b for empty working-set
    % Indices of the constraints in the working-set
    active = [];    % Indices for empty-working set

    while ~stop  % Continue until stop = 1
        % Initialize all mu as zero and update the mu in the working set
         mu0 = [0, 0];
        % Extract A corresponding to the working-set from A0
         A = A0(active, :);
        % Extract b corresponding to the working-set from b0
         b = b0(active);

        % Solve the QP problem given A and b
        [s, mu] = solve_activeset(x, W, c, A, b)
        % Round mu to prevent numerical errors (Keep this)
        mu = round(mu*1e12)/1e12
        %mu = round(mu);
        % Update mu values for the working-set using the solved mu values
         mu0(active) = mu;

        % Calculate the constraint values using the solved s values
         gcheck = A0*s-b0;
```

```
        % Round constraint values to prevent numerical errors (Keep this)
        gcheck = round(gcheck*1e12)/1e12
%         gcheck = round(gcheck)
        % Variable to check if all mu values make sense.
        mucheck = 0;        % Initially set to 0

        % Indices of the constraints to be added to the working set
        Iadd = [];              % Initialize as empty vector
        % Indices of the constraints to be added to the working set
        Iremove = [];           % Initialize as empty vector

        % Check mu values and set mucheck to 1 when they make sense
        if (numel(mu) == 0)
            % When there no mu values in the set
             mucheck = 1;         % OK
        elseif min(mu) > 0
            % When all mu values in the set positive
             mucheck = 1;         % OK
        else
            % When some of the mu are negative
            % Find the most negaitve mu and remove it from active set

             Iremove = find(mu==min(mu))  % Use Iremove to remove the constraint
            % Remove the index Iremove from the working-set
             active(Iremove) = [];
        end

        % Check if constraints are satisfied
        if max(gcheck) <= 0
            % If all constraints are satisfied
            if mucheck == 1
                % If all mu values are OK, terminate by setting stop = 1
                 stop = 1;
            end
        else
            % If some constraints are violated
            % Find the most violated one and add it to the working set
             Iadd = find(gcheck == max(gcheck)) % Use Iadd to add the constraint
            % Add the index Iadd to the working-set
             active(end+1) = Iadd
        end
        % Make sure there are no duplications in the working-set (Keep this)
        active = unique(active)
    end
end

function [s, mu] = solve_activeset(x, W, c, A, b)
    % Given an active set, solve QP

    % Create the linear set of equations given in equation (7.79)
    row = size(A);
    row = row(1);
     M = [W, A'; A, zeros(row)];
     U = [-1*c; b];
```

```
    sol = M\U;          % Solve for s and mu

     s = sol(1:length(x));                % Extract s from the solution
     mu = sol(length(x)+1:end);  % Extract mu from the solution
end
```