```
function [elk] = MD_estiff_1stnode_MyMz_release (A, Izz, Iyy, J, Ayy, Azz, E, v, L)
% Code developed by Mrunmayi Mungekar and Devasmit Dutta
%
% MD_estiff.m computes the element stiffness matrix for a given element with first
node flexurally released
%

%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  Functions Called
%              none
%
%  Dictionary of Variables
%  Input information
                % A = cross-sectional area
                % Izz = moment of inertia about local z-axis
                % Iyy = moment of inertia about local y-axis
                % J = torsional constant
                % Ayy = shear area along local y-axis
                % Azz = shear area along local z-axis
                % E = Young's modulus
                % v = Poisson's ratio
                % L = element length

                % G = shear modulus
                % elk_temp = temporary element stiffness matrix (just the lower
triangular part)
                % kA = axial stiffness
                % kJ = torsional stiffness
                % etaz = shear coefficient along local z-axis
                % etay = shear coefficient along local y-axis
%
% Output information
                % elk = complete element stiffness matrix
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%

% Consolidating the geometric and material properties

if(Izz == 0)
    Izz = Iyy;
elseif(Iyy == 0)
    Iyy = Izz;
elseif(J == 0)
    J = (Izz + Iyy)/10;
end
G = E / (2 + 2 * v);

elk_temp = zeros(12, 12);

kA = E * A / L;
kJ = G * J / L;
etaz = E * Iyy / (Azz * G);
etay = E * Izz / (Ayy * G);

%  Formulating lower half of the symmetric Kele
```

```
elk_temp(:, 1) = [kA; ...
    zeros(5,1); -kA;...
    zeros(5,1)];

elk_temp(:, 2) = E * Izz * [0; 1/4; ...
    zeros(3,1); 0; ...
    0; -1/4;...
    zeros(3,1); L / 4] / (L * (L ^ 2/12 + etay));

elk_temp(:, 3) = E * Iyy * [zeros(2, 1); 1/4;...
    0; 0; ...
    zeros(3,1); -1/4;...
    0; -L / 4;...
    0] / (L * (L ^ 2/12 + etaz));

elk_temp(:, 4) = [zeros(3, 1); kJ;...
    zeros(5,1); -kJ;...
    0; 0];

elk_temp(:, 5) = zeros(12, 1);

elk_temp(:, 6) = zeros(12,1);

elk_temp(:, 7) = [zeros(6, 1); kA;...
    zeros(5,1);];

elk_temp(:, 8) = E * Izz * [zeros(7, 1); 1/4;...
    zeros(3,1); -L / 4] / (L * (L ^ 2/12 + etay));

elk_temp(:, 9) = E * Iyy * [zeros(8, 1); 1/4;...
    0; -L / 4;...
    0] / (L * (L ^ 2/12 + etaz));

elk_temp(:, 10) = [zeros(9, 1); kJ;...
    0; 0];

elk_temp(:, 11) = E * Iyy * [zeros(10, 1); (L ^ 2/4 + etaz);...
    0] / (L * (L ^ 2/12 + etaz));

elk_temp(:, 12) = E * Izz * [zeros(11, 1); (L ^ 2/4 + etay)] / (L * (L ^ 2/12 +
etay));

% Inverting the lower half to form the entire symmetric matrix
[n, ~] = size(elk_temp);
elk = elk_temp' + elk_temp;
elk(1:n + 1:end) = diag(elk_temp);
```