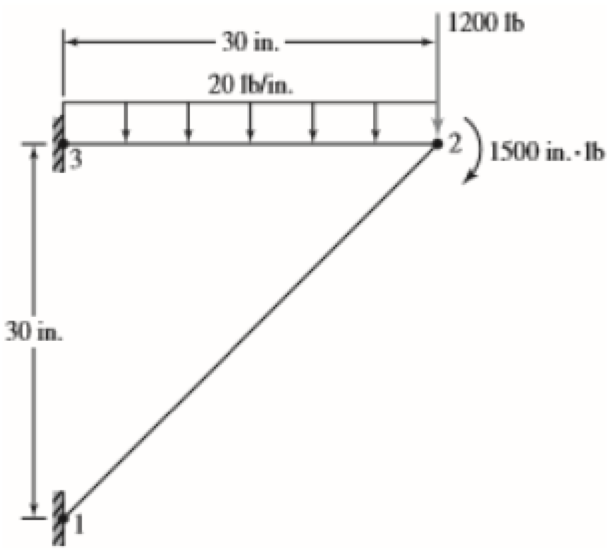
Problem 1

Use Mathcad, Matlab, ect to solve Logan Problems. Find the rections, nodal displacement and plot the interpolated displacement for the beams



The two-dimensional frame shown here is composed of two 2x4in. aluminum members ($E=10x10^6$psi). The 2in dimension is perpendicular to the plane of loading (in and out of the paper). All connections are treated as welded joints. Use the node numbers as shown.

Use Mathcad, Matlab, etc. to find:

a. The displacement components of node 2.
b. The reaction forces and moments at nodes 1 and 3.
c. The maximum stress in each element.

Part a:

In [1]:
```
# Import necessary packages
import copy
import numpy as np
import sympy as sp
sp.init_printing()
```

In [2]:
```
# Initialize problem parameters
E, I, A, L = sp.symbols('E, I, A, L')
```

In [3]:
```
# create stiffness matrix for element 1
# L = np.sqrt(2*(30**2))
C_1 = A*E/L
C_2 = E*I/(L**3)
```

In [4]:
```python
# Create pattern for local stiffness matrices
k_prime = sp.Matrix([[C_1, 0, 0, -C_1, 0, 0],
                     [0, 12*C_2, 6*C_2*L, 0, -12*C_2, 6*C_2*L],
                     [0, 6*C_2*L, 4*C_2*sp.Pow(L, 2), 0, -6*C_2*L, 2*C_2*sp.Po
w(L, 2)],
                     [-C_1, 0, 0, C_1, 0, 0],
                     [0, -12*C_2, -6*C_2*L, 0, 12*C_2, -6*C_2*L],
                     [0, 6*C_2*L, 2*C_2*sp.Pow(L, 2), 0, -6*C_2*L, 4*C_2*sp.Po
w(L, 2)]])
C, S = sp.symbols('C, S')
T = sp.Matrix([[C, S, 0, 0, 0, 0],
               [-S, C, 0, 0, 0, 0],
               [0, 0, 1, 0, 0, 0],
               [0, 0, 0, C, S, 0],
               [0, 0, 0, -S, C, 0],
               [0, 0, 0, 0, 0, 1]])
stiffness_pattern = sp.transpose(T)*k_prime*T
stiffness_pattern
```

Out[4]:

$$\begin{bmatrix} \frac{AC^2E}{L} + \frac{12EIS^2}{L^3} & \frac{ACES}{L} - \frac{12CEIS}{L^3} & -\frac{6EIS}{L^2} & -\frac{AC^2E}{L} - \frac{12EIS^2}{L^3} & -\frac{ACES}{L} + 1 \\ \frac{ACES}{L} - \frac{12CEIS}{L^3} & \frac{AES^2}{L} + \frac{12C^2EI}{L^3} & \frac{6CEI}{L^2} & -\frac{ACES}{L} + \frac{12CEIS}{L^3} & -\frac{AES^2}{L} - 1 \\ -\frac{6EIS}{L^2} & \frac{6CEI}{L^2} & \frac{4EI}{L} & \frac{6EIS}{L^2} & -\frac{6CEI}{L^2} \\ -\frac{AC^2E}{L} - \frac{12EIS^2}{L^3} & -\frac{ACES}{L} + \frac{12CEIS}{L^3} & \frac{6EIS}{L^2} & \frac{AC^2E}{L} + \frac{12EIS^2}{L^3} & \frac{ACES}{L} - 12 \\ -\frac{ACES}{L} + \frac{12CEIS}{L^3} & -\frac{AES^2}{L} - \frac{12C^2EI}{L^3} & -\frac{6CEI}{L^2} & \frac{ACES}{L} - \frac{12CEIS}{L^3} & \frac{AES^2}{L} + 12 \\ -\frac{6EIS}{L^2} & \frac{6CEI}{L^2} & \frac{2EI}{L} & \frac{6EIS}{L^2} & -\frac{6CEI}{L^2} \end{bmatrix}$$

In [5]:
```
# Construct local stiffness matrix for element 1
cross_sectional_area = 2*4
moment_of_inertia = 2*(4**3)/12
youngs_modulus = 10e6
element_length = np.sqrt(2*(30**2))
cos_ang = 30/element_length
sin_ang = cos_ang

k_1 = copy.deepcopy(stiffness_pattern)
k_1 = k_1.subs({A:cross_sectional_area,\
                I:moment_of_inertia,
                E:youngs_modulus,
                L:element_length,
                C:cos_ang,
                S:sin_ang})
k_1
```

Out[5]:

$$\begin{bmatrix} 951189.566396126 & 934428.516768 & -251415.744421884 & -951189.566 \\ 934428.516768 & 951189.566396126 & 251415.744421884 & -934428.5 \\ -251415.744421884 & 251415.744421884 & 10056629.7768753 & 251415.744 \\ -951189.566396126 & -934428.516768 & 251415.744421884 & 951189.566 \\ -934428.516768 & -951189.566396126 & -251415.744421884 & 934428.51 \\ -251415.744421884 & 251415.744421884 & 5028314.88843767 & 251415.744 \end{bmatrix}$$

In [6]:
```
# Construct local stiffness matrix for element 2
cos_ang = -1
sin_ang = 0
element_length = 30
k_2 = copy.deepcopy(stiffness_pattern)
k_2 = k_2.subs({A:cross_sectional_area,\
                I:moment_of_inertia,
                E:youngs_modulus,
                L:element_length,
                C:cos_ang,
                S:sin_ang})
k_2
```

Out[6]:

$$\begin{bmatrix} 2666666.66666667 & 0 & 0 & -2666666.66 \\ 0 & 47407.4074074074 & -711111.111111111 & 0 \\ 0 & -711111.111111111 & 14222222.2222222 & 0 \\ -2666666.66666667 & 0 & 0 & 2666666.666 \\ 0 & -47407.4074074074 & 711111.111111111 & 0 \\ 0 & -711111.111111111 & 7111111.11111111 & 0 \end{bmatrix}$$

In [7]:
```
# Construct global stiffness matrix
global_stiffness_matrix = sp.zeros(9)
global_stiffness_matrix[0:6,0:6] = k_1
# global_stiffness_matrix
global_stiffness_matrix[3:,3:] = global_stiffness_matrix[3:,3:] + k_2
global_stiffness_matrix
```

Out[7]:

$$\begin{bmatrix} 951189.566396126 & 934428.516768 & -251415.744421884 & -951189.566 \\ 934428.516768 & 951189.566396126 & 251415.744421884 & -934428.5 \\ -251415.744421884 & 251415.744421884 & 10056629.7768753 & 251415.744 \\ -951189.566396126 & -934428.516768 & 251415.744421884 & 3617856.23 \\ -934428.516768 & -951189.566396126 & -251415.744421884 & 934428.51 \\ -251415.744421884 & 251415.744421884 & 5028314.88843767 & 251415.744 \\ 0 & 0 & 0 & -2666666.66 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

In [8]:
```
# Reduce the global stiffness matrix from what is constrained
reduced_stiffness_matrix = global_stiffness_matrix[3:6, 3:6]
reduced_stiffness_matrix
```

Out[8]:

$$\begin{bmatrix} 3617856.23306279 & 934428.516768 & 251415.744421884 \\ 934428.516768 & 998596.973803533 & -962526.855532995 \\ 251415.744421884 & -962526.855532995 & 24278851.9990976 \end{bmatrix}$$

In [9]:
```
# Construct reduced force matrix
reduced_force_matrix = sp.Matrix([0, -1500, 0])
reduced_force_matrix
```

Out[9]:

$$\begin{bmatrix} 0 \\ -1500 \\ 0 \end{bmatrix}$$

In [10]:
```
# Solve for unconstrained displacements at node 2
non_constrained_displacements = reduced_stiffness_matrix.LUsolve(reduced_force
_matrix)
non_constrained_displacements
```

Out[10]:

$$\begin{bmatrix} 0.000548994474615222 \\ -0.00210161222558867 \\ -8.90027280385732 \cdot 10^{-5} \end{bmatrix}$$

Part b:

In [11]:
```
# Solve for all loads/reactions
displacements = sp.Matrix([0, 0, 0, non_constrained_displacements, 0, 0, 0])
equivalent_nodal_forces = sp.Matrix([0, 0, 0, 0, 0, 0, 0, -300, -1500])
F = sp.N(global_stiffness_matrix*displacements, 5) - equivalent_nodal_forces
print("Global Force Matrix:")
F
```

```
Global Force Matrix:
```

Out[11]:

$$\begin{bmatrix} 1464.0 \\ 1463.7 \\ 218.87 \\ -2.8422 \cdot 10^{-14} \\ -1500.0 \\ -4.5475 \cdot 10^{-13} \\ -1464.0 \\ 336.34 \\ 2361.6 \end{bmatrix}$$

Part c:

In [12]:
```
# Calculate local forces of element 1
element_length = np.sqrt(2*(30**2))
element_displacements_1 = sp.Matrix(displacements[0:6])
k_prime_1 = copy.deepcopy(k_prime)
k_prime_1 = k_prime_1.subs({A:cross_sectional_area,\
                            E:youngs_modulus,
                            I:moment_of_inertia,
                            L:element_length})
cos_ang = 30/element_length
sin_ang = cos_ang
rotation_matrix_1 = copy.deepcopy(T)
rotation_matrix_1 = rotation_matrix_1.subs({C:cos_ang,
                                            S:sin_ang})
local_forces_1 = k_prime_1*rotation_matrix_1*element_displacements_1
local_forces_1
```

Out[12]:

$$\begin{bmatrix} 2070.15700129793 \\ -0.230816485372475 \\ 218.870514193468 \\ -2070.15700129793 \\ 0.230816485372475 \\ -228.663228314459 \end{bmatrix}$$

In [13]:
```python
# Choose greatest axial load in the element and calculate the greatest axial s
tress in the element
greatest_axial = np.abs(local_forces_1[0])/cross_sectional_area

# Choose greates bending moment in the element and calculate the greatest norm
al stress due to bending
greatest_bending = np.abs(local_forces_1[-1])*2/moment_of_inertia

# Add the greatest axial stress and normal stress due to bending
print("max stress in element 1: {} psi".format(greatest_axial + greatest_bendi
ng))
```

```
max stress in element 1: 301.643980471203 psi
```

In [14]:
```python
# Calculate local forces of element 2
element_length = 30
element_displacements_2 = sp.Matrix(displacements[3:])
k_prime_2 = copy.deepcopy(k_prime)
k_prime_2 = k_prime_2.subs({A:cross_sectional_area,\
                            E:youngs_modulus,
                            I:moment_of_inertia,
                            L:element_length})
cos_ang = -1
sin_ang = 0
rotation_matrix_2 = copy.deepcopy(T)
rotation_matrix_2 = rotation_matrix_2.subs({C:cos_ang,
                                            S:sin_ang})
local_forces = k_prime_2*rotation_matrix_2*element_displacements_2
local_forces
```

Out[14]:

$$\begin{bmatrix} -1463.98526564059 \\ 36.3411581634405 \\ 228.663228314459 \\ 1463.98526564059 \\ -36.3411581634405 \\ 861.571516588757 \end{bmatrix}$$

In [15]:
```python
# Choose greated axial load in the element and calculate the greatest axial st
ress in the element
greatest_axial = np.abs(local_forces[0])/cross_sectional_area

# Choose greates bending moment in the element and calculate the greatest norm
al stress due to bending
greatest_bending = np.abs(local_forces[-1])*2/moment_of_inertia

# Add the greatest axial stress and normal stress due to bending
print("max stress in element 2: {} psi".format(greatest_axial + greatest_bendi
ng))
```

```
max stress in element 2: 344.542817565466 psi
```

In [ ]: