#1

a)

```
ABD =    [[ 1.024e+08   1.894e+07   0           0           0           0        ]
          [ 1.894e+07   1.625e+07   0           0           0           0        ]
          [ 0           0           2.019e+07   0           0           0        ]
          [ 0           0           0           5.779e+00   1.766e+00   1.261e+00]
          [ 0           0           0           1.766e+00   1.256e+00   4.177e-01]
          [ 0           0           0           1.261e+00   4.177e-01   1.850e+00]]
```

```
abd =    [[ 1.245e-08  -1.452e-08   0           0           0           0        ]
          [-1.452e-08   7.846e-08   0           0           0           0        ]
          [ 0           0           4.953e-08   0           0           0        ]
          [ 0           0           0           3.299e-01  -4.205e-01  -1.299e-01]
          [ 0           0           0          -4.205e-01   1.397e+00  -2.873e-02]
          [ 0           0           0          -1.299e-01  -2.873e-02   6.357e-01]]
```

b)

$$\bar{\upsilon}_{xy} = 1.1658$$

c) Very high effective Poisson's ratio, strain in the transverse direction will be much higher than in the axial direction. This is the benefit of composites since most base materials have an upper limit of .5 for their Poisson's ratio.

Source for common values: https://www.engineeringtoolbox.com/poissons-ratio-d_1224.html

d)

```
                    [[[  92798.759]
                      [  30066.954]
                      [  46706.313]]

                     [[  92798.759]
                      [  30066.954]
                      [ -46706.313]]

                     [[ 155747.789]
                      [   3015.277]
sigma_xyz = e   *     [      0.   ]]

                     [[ 155747.789]
                      [   3015.277]
                      [      0.   ]]

                     [[  92798.759]
                      [  30066.954]
                      [ -46706.313]]

                     [[  92798.759]
                      [  30066.954]
                      [  46706.313]]]
```

#3

$$\begin{bmatrix} \hat{N}_x^T \\ \hat{N}_y^T \\ \hat{N}_{xy}^T \end{bmatrix} = \begin{bmatrix} 1.85778247e+02 \\ 2.53256191e+02 \\ -1.98294846e-14 \end{bmatrix} {}^{N}/_{°C}$$

$$\begin{bmatrix} \hat{M}_x^T \\ \hat{M}_y^T \\ \hat{M}_{xy}^T \end{bmatrix} = \begin{bmatrix} -5.20417043e-18 \\ -3.46944695e-18 \\ -1.64208032e-18 \end{bmatrix} {}^{N}/_{m\,°C}$$

```
import numpy as np

def Transform(theta):
    m = np.cos( np.deg2rad(theta) )
    n = np.sin( np.deg2rad(theta) )
    return np.array([
        [m**2, n**2, 2*m*n],
        [n**2, m**2, -2*m*n],
        [-m*n, m*n,  m**2 - n**2]], np.float64)

theta = np.array([30,-30,0,0,-30,30])

N = theta.size
h = .15*10**-3
H = N*h

Z = np.arange(N+1)*h - .5*H

E1 = 155 * 10**9
E2 = 12.1 * 10**9
v12 = .248
G12 = 4.4 * 10**9

S = np.array([
    [1/E1, -v12/E1, 0],
    [-v12/E1, 1/E2, 0],
    [0,  0       , 1/G12]], np.float64)

T = Transform(theta)
T_ = np.rollaxis(T, 2)

Sbar = np.einsum('...jk,kl,...lm->...jm', T.T, S, T_)
Qbar = np.linalg.inv(Sbar)

A = np.sum( np.diff(Z)[:, None, None] * Qbar, axis=0)
B = (1/2)*np.sum( np.diff(Z**2)[:, None, None] * Qbar, axis=0)
D = (1/3)*np.sum( np.diff(Z**3)[:, None, None] * Qbar, axis=0)

ABD = np.vstack( (np.hstack((A,B)), np.hstack((B,D))) )
ABD[np.abs(ABD) < 10**-8] = 0

abd = np.linalg.inv(ABD)

nu_bar_xy = -(abd[0,1]/abd[0,0])

strain = np.array([[10**-6, 0, 0]]).T
stress = np.matmul(Qbar, strain)
```

```
import numpy as np

#np.set_printoptions(precision=4)

def Transform(theta):
    m = np.cos( np.deg2rad(theta) )
    n = np.sin( np.deg2rad(theta) )
    return np.array([
        [m**2, n**2, 2*m*n],
        [n**2, m**2, -2*m*n],
        [-m*n, m*n,  m**2 - n**2]], np.float64)

theta = np.array([0,90,+30,-30,-30,+30,90,0])

N = theta.size
h = 150*10**-6
H = N*h

Z = np.arange(N+1)*h - .5*H

alpha1 = -.018*10**-6
alpha2 = 24.3*10**-6
alpha3 = alpha2

alpha = np.array([[alpha1, alpha2, 0]]).T

E1 = 155 * 10**9
E2 = 12.1 * 10**9
v12 = .248
G12 = 4.4 * 10**9

S = np.array([
    [1/E1, -v12/E1, 0],
    [-v12/E1, 1/E2, 0],
    [0,  0       , 1/G12]], np.float64)

T = Transform(theta)
T_ = np.rollaxis(T, 2)

alpha_bar = np.matmul(T.T, alpha)

Sbar = np.einsum('...jk,kl,...lm->...jm', T.T, S, T_)
Qbar = np.linalg.inv(Sbar)

N_t = np.sum( np.diff(Z)[:, None, None] * np.matmul(Qbar, alpha_bar), axis=0 )
M_t = (1/2)*np.sum( np.diff(Z**2)[:, None, None] * np.matmul(Qbar, alpha_bar),
axis=0 )
```