Question 1

a) $$\begin{bmatrix} sx' \\ sy' \\ s \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\left.\begin{aligned} sx' &= a_1x + a_2y + a_3 \\ sy' &= a_4x + a_5y + a_6 \\ s &= a_7x + a_8y + a_9 \end{aligned}\right\} \rightarrow \quad x' = \frac{a_1x + a_2y + a_3}{a_7x + a_8y + a_9} \qquad y' = \frac{a_4x + a_5y + a_6}{a_7x + a_8y + a_9}$$

$$(a_7x + a_8y + a_9) \cdot x' = (a_1x + a_2y + a_3)$$
$$(a_7x + a_8y + a_9) \cdot y' = (a_4x + a_5y + a_6)$$

$$(a_1x + a_2y + a_3) - (a_7x + a_8y + a_9) \cdot x' = 0$$
$$(a_4x + a_5y + a_6) - (a_7x + a_8y + a_9) \cdot y' = 0$$

with the n pairs we can write the equations as:

$$a_1x_1 + a_2y_1 + a_3 - (a_7x_1 + a_8y_1 + a_9) \cdot x_1' = 0$$
$$a_4x_1 + a_5y_1 + a_6 - (a_7x_1 + a_8y_1 + a_9) \cdot y_1' = 0$$
$$\vdots$$
$$a_1x_n + a_2y_n + a_3 - (a_7x_n + a_8y_n + a_9) \cdot x_n' = 0$$
$$a_4x_n + a_5y_n + a_6 - (a_7x_n + a_8y_n + a_9) \cdot y_n' = 0$$

2

$$
\begin{array}{ccccccccc}
a_1 & a_2 & a_3 & a_4 & & & & & \\
\end{array}
$$

$$
\begin{bmatrix}
x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1 \cdot x_1' & -y_1 \cdot x_1' & -x_1' \\
0 & 0 & 0 & x_1 & y_1 & 1 & -x_1 \cdot y_1' & -y_1 \cdot y_1' & -y_1' \\
 & & & & \vdots & & & & \\
x_n & y_n & 1 & 0 & 0 & 0 & -x_n \cdot x_n' & -y_n \cdot x_n' & -x_n' \\
0 & 0 & 0 & x_n & y_n & 1 & -x_n \cdot y_n' & -y_n \cdot y_n' & -y_n'
\end{bmatrix}
$$

$$\Downarrow$$

Q matrix, $a = \begin{bmatrix} a_1 \\ \vdots \\ a_9 \end{bmatrix}$

So, we wrote the

$Q\,a = 0$

as

$$
\begin{array}{ccccccccc}
a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 & a_8 & a_9
\end{array}
$$

$$
\underset{2n \times 9}{\begin{bmatrix}
x_1 & y_1 & 1 & 0 & 0 & 0 & (-x_1 x_1') & (-y_1 x_1') & (-x_1') \\
0 & 0 & 0 & x_1 & y_1 & 1 & (-x_1 y_1') & (-y_1 y_1') & (-y_1') \\
 & & & & & & & & \\
x_n & y_n & 1 & 0 & 0 & 0 & -x_n x_n' & -y_n x_n' & -x_n' \\
0 & 0 & 0 & x_n & y_n & 1 & -x_n y_n' & -y_n y_n' & -y_n'
\end{bmatrix}}
\underset{9 \times 1}{\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_9 \end{bmatrix}}
=
\underset{2n \times 1}{\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}}
$$

(b) with the given numerical values the Q matrix come at as follows:

$$Q = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -5 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -4 \\ 1 & 0 & 1 & 0 & 0 & 0 & -7 & 0 & -7 \\ 0 & 0 & 0 & 1 & 0 & 1 & -4 & 0 & -4 \\ 1 & 1 & 1 & 0 & 0 & 0 & -7 & -7 & -7 \\ 0 & 0 & 0 & 1 & 1 & 1 & -5 & -5 & -5 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & -6 & -6 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & -6 & -6 \end{bmatrix}$$

we need to find the matrix $Q^T \cdot Q$:

```
[1]  import numpy as np
```

```
Q = [[0, 0, 1, 0, 0, 0, 0, 0, -5],
     [0, 0, 0, 0, 0, 1, 0, 0, -4],
     [1, 0, 1, 0, 0, 0,-7, 0, -7],
     [0, 0, 0, 1, 0, 1,-4, 0, -4],
     [1, 1, 1, 0, 0, 0,-7, -7, -7],
     [0, 0, 0, 1, 1, 1,-5, -5, -5],
     [0, 1, 1, 0, 0, 0, 0, -6, -6],
     [0, 0, 0, 0, 1, 1, 0, -6, -6]]
```

```
[9]   QT = np.transpose(Q)
      QT
```

```
array([[ 0,  0,  1,  0,  1,  0,  0,  0],
       [ 0,  0,  0,  0,  1,  0,  1,  0],
       [ 1,  0,  1,  0,  1,  0,  1,  0],
       [ 0,  0,  0,  1,  0,  1,  0,  0],
       [ 0,  0,  0,  0,  0,  1,  0,  1],
       [ 0,  1,  0,  1,  0,  1,  0,  1],
       [ 0,  0, -7, -4, -7, -5,  0,  0],
       [ 0,  0,  0,  0, -7, -5, -6, -6],
       [-5, -4, -7, -4, -7, -5, -6, -6]])
```

Figure 1: Q and Q transpose matrix

In Figure 2, we multiply the Q and Q transpose and we input the resulting matrix to function np.linalg.eig. The function gives two outputs. It gives a matrix(Figure 3) of eigenvectors(in columns) and gives a 1-D array of corresponding eigenvalues of the eigenvectors. Therefore the smallest eigenvalue of the 1-D array indicates the solution for the ‘a’ vector.

```
[10] Symmetric_matrix = np.matmul(QT,Q)
     Symmetric_matrix
```

```
array([[  2,   1,   2,   0,   0,   0, -14,  -7, -14],
       [  1,   2,   2,   0,   0,   0,  -7, -13, -13],
       [  2,   2,   4,   0,   0,   0, -14, -13, -25],
       [  0,   0,   0,   2,   1,   2,  -9,  -5,  -9],
       [  0,   0,   0,   1,   2,   2,  -5, -11, -11],
       [  0,   0,   0,   2,   2,   4,  -9, -11, -19],
       [-14,  -7, -14,  -9,  -5,  -9, 139,  74, 139],
       [ -7, -13, -13,  -5, -11, -11,  74, 146, 146],
       [-14, -13, -25,  -9, -11, -19, 139, 146, 252]])
```

```
[13] eigenvalues, matrix = np.linalg.eigh(Symmetric_matrix)
```

```
[15] np.set_printoptions(suppress=True)
     print(eigenvalues)
```

```
[ -0.           0.00879951   0.01307177   0.62936395   1.04621443
   6.32190546  32.58386497  69.45979685 442.93698306]
```

Figure 2: np.linalg.eigh function is used

```
##smallest eigenvalue is the 1st element, hence the eigenvector that gives the smallest eigenvalue
## of the matrix below is the first column. That will be the solution for vector a.
print(matrix)
```

```
[[ 0.73029674  0.31569415  0.26671749 -0.31742916  0.35241227 -0.24670845
   0.04362755  0.07758551 -0.04606554]
 [ 0.31950483 -0.68770976 -0.05304357 -0.30684075 -0.48559227 -0.29023778
   0.0542431  -0.0558555  -0.04367046]
 [ 0.22821773  0.27483107 -0.66779499  0.41905063 -0.16107061 -0.46039615
  -0.08676274  0.01785284 -0.07124691]
 [ 0.36514837  0.2239834   0.35793336  0.41373021 -0.60504439  0.38348674
   0.03537917  0.0436298  -0.03021729]
 [ 0.36514837 -0.50952875 -0.14969644  0.47013732  0.48558576  0.34954222
   0.03057906 -0.05839005 -0.0359336 ]
 [ 0.18257419  0.16238127 -0.55651679 -0.48708554 -0.10782779  0.61072023
  -0.07455756 -0.01710941 -0.05364835]
 [ 0.09128709  0.04263619  0.06383694  0.0008212  -0.00945497 -0.02245244
  -0.50341454 -0.71957402  0.46256514]
 [ 0.04564355 -0.10826417  0.00093936  0.00241872 -0.00795203  0.01332353
  -0.53224529  0.68347859  0.48530405]
 [ 0.04564355  0.04653609 -0.12564127  0.00483949 -0.02025528  0.00544928
   0.66570993  0.00254525  0.73234265]]
```

Figure 3: Eigenvectors

```
[35] col1 = [val[0] for val in matrix]
     print(col1)

     [0.7302967433412046, 0.319504825209886, 0.22821773229355038, 0.365148371670974, 0.365148371668889, 0.18257418583472038, 0.09128709291769165, 0.0456435464585

[36] ##normalize by a9
     for i in range(9):
       col1[i]=col1[i]/(0.045643546458704956)
     print(col1)
     ##the resulting vector is 'a'.

     [16.00000000004218, 6.999999999977024, 5.000000000000561, 8.000000000029235, 7.9999999999835545, 3.9999999999978213, 2.0000000000061724, 0.9999999999964964,
```

Figure 4: The solution vector

The normalized solution vector is:

```
[16.00000000004218, 6.999999999977024, 5.000000000000561, 8.000000000029235,
7.9999999999835545, 3.9999999999978213, 2.0000000000061724, 0.9999999999964964,
1.0]
```