My student ID Card number last two digit are 73

B = 7

A = 3

I have use the numerical values for Q1 and Q 6

Q1) A

Q1) a)

(i) $C^2 =$

$$C = \begin{pmatrix} 3 & 2 & -5 & 4 \\ 0 & -4 & 7 & 8 \\ 4 & 3 & 2 & 5 \\ 3 & 1 & 0 & -7 \end{pmatrix}^2$$

$$\begin{pmatrix} 3 & 2 & -5 & 4 \\ 0 & -4 & 7 & 8 \\ 4 & 3 & 2 & 5 \\ 3 & 1 & 0 & -7 \end{pmatrix} \begin{pmatrix} 3 & 2 & -5 & 4 \\ 0 & -4 & 7 & 8 \\ 4 & 3 & 2 & 5 \\ 3 & 1 & 0 & -7 \end{pmatrix}$$

$\Big((3\times3)+(2\times0)+(-5\times4)+(4\times3) \quad (3\times2)+(2\times-4)+(-5\times3)+(4\times1) \quad (3\times-5)+(2\times7)+$

$\downarrow$

$$
\begin{pmatrix}
(3\times3)+(2\times0)+(-5\times4)+(4\times3) & (3\times2)+(2\times-4)+(-5\times3)+(4\times1) & (3\times-5)+(2\times7)+(-5\times2)+(4\times0) & (3\times4)+(2\times8)+(-5\times5)+(4\times-7) \\
(0\times3)+(-4\times0)+(7\times4)+(8\times3) & (0\times2)+(-4\times-4)+(7\times3)+(8\times1) & (0\times-5)+(-4\times7)+(7\times2)+(8\times0) & (0\times4)+(-4\times8)+(7\times5)+(8\times-7) \\
(4\times3)+(3\times0)+(2\times4)+(5\times3) & (4\times2)+(3\times-4)+(2\times3)+(5\times1) & (4\times-5)+(3\times7)+(2\times2)+(5\times0) & (4\times4)+(3\times8)+(2\times5)+(5\times-7) \\
(3\times3)+(1\times0)+(0\times4)+(-7\times3) & (3\times2)+(1\times-4)+(0\times3)+(-7\times1) & (3\times-5)+(1\times7)+(0\times2)+(-7\times0) & (3\times4)+(1\times8)+(0\times5)+(-7\times-7)
\end{pmatrix}
$$

$$
= \begin{pmatrix}
9+0-20+12 & 6-8-15+4 & -15+14-10+0 & 12+16-25-28 \\
0-0+28+24 & 0+16+21+8 & 0-28+14+0 & 0-32+35-56 \\
12+0+8+15 & 8-12+6+5 & -20+21+4+0 & 16+24+10-35 \\
9+0+0-21 & 6-4+0-7 & -15+7+0+0 & 12+8+0+49
\end{pmatrix}
$$

$$
= \begin{pmatrix}
1 & -13 & -11 & -25 \\
52 & 45 & -14 & -53 \\
35 & 7 & 5 & 15
\end{pmatrix}
$$

ANSWER =

Answer

$$= \begin{bmatrix} 1 & -13 & -11 & -25 \\ 52 & 45 & -14 & -53 \\ 35 & 7 & 5 & 15 \\ -12 & -5 & 8 & 69 \end{bmatrix} //$$

(iv) $(3C - D)\vec{v}$

$$= 3\begin{pmatrix} 3 & 2 & -5 & 4 \\ 0 & -4 & 7 & 8 \\ 4 & 3 & 2 & 5 \\ 3 & 1 & 0 & -7 \end{pmatrix} = \begin{pmatrix} 9 & 6 & -15 & 12 \\ 0 & -12 & 21 & 24 \\ 12 & 9 & 6 & 15 \\ 9 & 3 & 0 & -21 \end{pmatrix}$$

$$\begin{pmatrix} 3\times 3 & 3\times 2 & 3\times -5 & 3\times 4 \\ 3\times 0 & 3\times -4 & 3\times 7 & 3\times 8 \\ 3\times 4 & 3\times 3 & 3\times 2 & 3\times 5 \\ 3\times 3 & 3\times 1 & 3\times 0 & 3\times -7 \end{pmatrix}$$

Lets solve for

$(3C - D)$

$$\begin{pmatrix} 9 & 6 & -15 & 12 \\ 0 & -12 & 21 & 24 \\ & 9 & 6 & 15 \end{pmatrix} - \begin{pmatrix} -5 & 3 & -1 & 2 \\ 4 & 2 & 1 & -5 \\ 3 & 7 & -7 & -3 \\ 1 & 0 & 6 & 4 \end{pmatrix}$$

Lets solve for

$(3C - D)$

$$\begin{pmatrix} 9 & 6 & -15 & 12 \\ 0 & -12 & 21 & 24 \\ 12 & 9 & 6 & 15 \\ 9 & 3 & 0 & -21 \end{pmatrix} - \begin{pmatrix} -5 & 3 & -1 & 2 \\ 4 & 2 & 1 & -5 \\ 3 & 7 & -7 & -3 \\ 1 & 0 & 6 & 4 \end{pmatrix}$$

$$\begin{bmatrix} (9)-(-5) & (6)-(3) & (-15)-(-1) & (12-(2)) \\ (0)-(4) & (-12)-(2) & (21)-(1) & (24)-(-5) \\ (12)-(3) & (9)-(7) & (6)-(-7) & (15-(-3)) \\ (9)-(1) & (3)-(0) & (0)-(6) & (-21-(4)) \end{bmatrix}$$

$$\begin{bmatrix} 14 & 3 & -14 & 10 \\ -4 & -14 & 20 & 29 \\ 9 & 2 & 13 & 18 \\ 8 & 3 & -6 & -25 \end{bmatrix}$$

Solve for

$(3C - D)\vec{v}$

$$\begin{bmatrix} 14 & 3 & -14 & 10 \\ -4 & -14 & 20 & 29 \\ 9 & 2 & 13 & 18 \\ 8 & 3 & -6 & -25 \end{bmatrix} \times \begin{bmatrix} 3 \\ 3 \\ -7 \\ -7 \end{bmatrix}$$

(ii) $(3C - D)\vec{v}$

Dimension mismatch.

(iii) $\vec{v}\vec{v}^T$

$$\begin{bmatrix} 3 \\ 3 \\ -7 \\ 7 \end{bmatrix} \begin{bmatrix} 3 & 3 & -7 & 7 \end{bmatrix}$$

$$\begin{bmatrix} (3)\times(3) & (3)\times(3) & (3)\times(-7) & (3)\times(7) \\ (3)\times(3) & (3)\times(3) & (3)\times(-7) & (3)\times(7) \\ (-7)\times(3) & (-7)\times(3) & (-7)\times(-7) & (-7)\times(7) \\ (7)\times(3) & (7)\times(3) & (7)\times(-7) & (7)\times(7) \end{bmatrix}$$

$$= \begin{bmatrix} 9 & 9 & -21 & 21 \\ 9 & 9 & -21 & 21 \\ -21 & -21 & 49 & -49 \\ 21 & 21 & -49 & 49 \end{bmatrix}$$

//.

#Q1b) Answer

```
#Q1b) Answer
# Importing the required library
import numpy as np

# Defining the matrix E and vector b
matrix_E = np.array([[-1, 0, -10, 6],
                     [2, 3, 5, 1],
                     [4, 2, 2, -2],
                     [0, 1, -3, 5]])

vector_b = np.array([26, 7, 2, 23])

# Calculating the inverse of matrix E
inverse_E = np.linalg.inv(matrix_E)

# Solving for vector x using the inverse: x = inverse_E * vector_b
solution_x = np.dot(inverse_E, vector_b)

# Displaying the solution for vector x
print("Solution for vector x:")
print(solution_x)
```

```
Solution for vector x:
[ 4. -2.  0.  5.]
```

```
# Importing the required library

import numpy as np


# Defining the matrix E and vector b

matrix_E = np.array([[-1, 0, -10, 6],

                     [2, 3, 5, 1],

                     [4, 2, 2, -2],

                     [0, 1, -3, 5]])
```

```
vector_b = np.array([26, 7, 2, 23])


# Calculating the inverse of matrix E
inverse_E = np.linalg.inv(matrix_E)


# Solving for vector x using the inverse: x = inverse_E * vector_b
solution_x = np.dot(inverse_E, vector_b)


# Displaying the solution for vector x
print("Solution for vector x:")
print(solution_x)
```

C)

Q1) c)

To calculate the length of vector $\vec{x}_{true} - \vec{x}_{approx.}$

1) Deduct the elements of vectors $\vec{x}_{true}$ and $\vec{x}_{approx.}$
2) Square each result in step 1
3) Get the total of squared values
4) Square root of total obtained in step 3

Vectors as follows:-

$\vec{x}_{true} = [50, 81, 86, 87, 99]$

$\vec{x}_{approx} = [56, 78, 85, 90, 95]$

Calculate the subtraction. & do the steps above

$x_{true} = [50, 81, 86, 87, 99]$

$\vec{x}_{approx} = [56, 78, 85, 90, 95]$

Calculate the subtraction. & do the steps above

1) $\vec{d} = \vec{x}_{true} - \vec{x}_{approx} = [50-56, 81-78, 86-85, 87-90, 99-95]$

$= [-6, 3, 1, -3, 4]$

2) Square each element $\vec{d}$: $\vec{d}^2 = [36, 9, 1, 9, 16]$

3) Total of squared values : $36 + 9 + 1 + 9 + 16 = 71$

4) square root of total : $\sqrt{71}$

$\therefore$ so lenght of vector $\vec{x}_{true} - \vec{x}_{approx}$ is $\sqrt{71}$ approximat

The square root of 71 is approximately 8.42615.

Q2 A)

Q2)

(a) $\begin{bmatrix} 3 & 2 \\ 1 & 2 \end{bmatrix}$

To begin, subtract λ from the diagonal entries of the provided matrix to create a new matrix:

$$\begin{bmatrix} 3-\lambda & 2 \\ 1 & 2-\lambda \end{bmatrix}$$

The determinant of 2x2 matrix is

$$\begin{bmatrix} 3-\lambda & 2 \\ 1 & 2-\lambda \end{bmatrix} = (3-\lambda)\times(2-\lambda) - (2)\times(1) = \lambda^2 - 5\lambda + 4$$

$$(\lambda - 4)(\lambda - 1) = 0$$

The roots are $\lambda_1 = 4$, $\lambda_2 = 1$

These are the eigenvalues

Next eigenvectors

$\lambda = 4$

$$\begin{bmatrix} 3-\lambda & 2 \\ 1 & 2-\lambda \end{bmatrix} = \begin{bmatrix} -1 & 2 \\ 1 & -2 \end{bmatrix}$$

$\downarrow$ reduced row echelon form $\begin{bmatrix} -1 & 2 \\ 1 & -2 \end{bmatrix}$

Multiply row 1 by -1 : $R_1 = -R_1$

$$\begin{bmatrix} +1 & -2 \\ 1 & -2 \end{bmatrix}$$

Subtract row 1 from row 2 ; $R_2 = R_2 - R_1$

$\begin{bmatrix} 1 & -2 \\ 0 & 0 \end{bmatrix}$ ← reduced row echelon form

To find the null s[Open with] the matrix equation

$$\begin{bmatrix} 1 & -2 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

if we take $x_2 = t$ and $x_1 = 2t$

Thus $\vec{x} = \begin{bmatrix} 2t \\ t \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix} t$

this is null space

Nullity of matrix is the dimension of the basis for null space

Thus nullity of matrix is 1

$$\begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

This is the eigenvector

- $\lambda = 1$

$$\begin{bmatrix} 3-\lambda & 2 \\ 1 & 2-\lambda \end{bmatrix} = \begin{bmatrix} 2 & 2 \\ 1 & 1 \end{bmatrix}$$

~~Null space of matrix~~ $\cancel{\begin{bmatrix} -1 \\ 1 \end{bmatrix}}$

~~reduced row echelon form of matrix~~ $\begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}$

reduce row echelon form of $\begin{bmatrix} 2 & 2 \\ 1 & 1 \end{bmatrix}$

Divide row 1 by 2 $R_1 = \frac{R_1}{2}$

$$\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

~~Subtract row~~

Subtract row 1 from row 2 : $R_2 = R_2 - R_1$ $\begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}$

reduced row echelon form $\begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}$

Find null space solve matrix $\begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

if we take $x_2 = t$ and $x_1 = -t$

Thus $\vec{x} = \begin{bmatrix} -t \\ t \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \end{bmatrix} t$

This is null space

nullity of a matrix is the dimension basis for null space.

Thus nullity of matrix is 1

$\begin{bmatrix} -1 \\ 1 \end{bmatrix}$

Answer.

Eigen value: 4, multiplicity: 1, eigenvector $\begin{bmatrix} 2 \\ 1 \end{bmatrix}$

Eigen value: 1, multiplicity: 1, eigenvector $\begin{bmatrix} -1 \\ 1 \end{bmatrix}$

(b) Lets demonstrate that for an $n \times n$ matrix A and an eigen vector $\bar{x}$

Q2 B)

(b)

To show that for the matrix A and an eigenvector x with eigen value λ, $A^K x = \lambda^K x$ for $K \in N$, $K > 1$:

The base case $K = 2$:

$$A^2 x = A(Ax) = A(\lambda x) = \lambda(Ax) = \lambda(\lambda x) = \lambda^2 x$$

~~Now assume the statement is true for K = m, where m > 1:~~

Now assume the statement is true for $K = m$, where $m > 1$:

$$A^m x = \lambda^m x$$

multiply both sides by A:

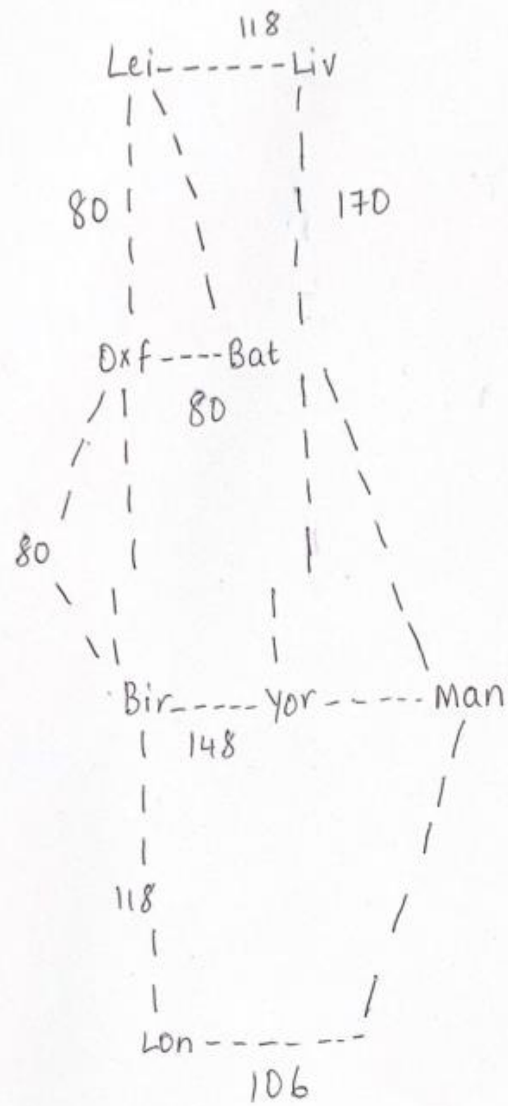
$$A^{m+1} x = A(\lambda^m x) = \lambda^m (Ax) = \lambda^m (\lambda x) = \lambda^{m+1} x$$

Therefore by mathematical induction, $A^K x = \lambda^K x$ for all $K \in N$, $K > 1$

Q3 A),B,D,E ANSWERS

Q3)

(a) Drawing the weighted graph G:



An example of a cycle in G is ~~Lon $\to$ Lei $\to$ Liv $\to$ Oxf~~

Lon $\to$ Lei $\to$ Liv $\to$ Oxf $\to$ Bat $\to$ Lon and its length is

$118 + 118 + 170 + 80 + 80 = 566$ miles.

Q3) The adjacency matrix for G.
b)

	Lon	Lei	Yor	Man	Liv	Bir	Oxf	Bat
Lon	0	106	211	0	0	118	0	118
Lei	106	0	0	0	118	0	80	0
Yor	211	0	0	80	0	148	0	0
Man	0	0	80	0	0	0	0	0
Liv	0	118	0	0	0	0	0	0
Bir	118	0	148	0	0	0	0	0
Oxf	0	80	0	0	0	0	0	80
Bat	118	0	0	0	0	0	80	0

In the adjacency matrix:

- The entry at the i-th row and j-th column represents the weight of the edge between vertex i and vertex j.
- If there is direct edge between the vertices, the entry is 0

The diagonal entries are all 0 because there are no self-loops in graph

Q3
d) (i) Sort the edges by weight.

(Man, Yor) 80
(Oxf, Lei) 80
(Oxf, Bat) 80
(Lon, Bir) 118
(Lon, Bat) 118
(Lei, Liv) 118
(Yor, Bir) 148
(Liv, Oxf) 170

2. Build the minimum spanning Tree:

- Add (Man, Yor) 80
- Add (Oxf, Lei) 80
- Add (Oxf, Bat) 80
- Add (Lon, Bir) 118
- Add (Lei, Liv) 118

Step	Edge	MST Edges	MST Vertices	Total Distance
1	(Man, Yor)	(Man, Yor)	{Man, Yor}	80
2	(Oxf, Lei)	(Man, Yor), (Oxf, Lei)	{Man, Yor, Lei, Oxf}	160
3	(Oxf, Bat)	(Man, Yor), (Oxf, Lei), (Oxf, Bat)	{Man, Yor, Lei, Oxf, Bat}	240
4	(Lon, Bir)	(Man, Yor), (Oxf, Lei), (Oxf, Bat), (Lon, Bir)	{Man, Yor, Lei, Oxf, Bat, Lon, Bir}	358
5	(Lei, Liv)	(Man, Yor), (Oxf, Lei), (Oxf, Bat), (Lon, Bir), (Lei, Liv)	{Man, Yor, Lei, Oxf, Bat, Lon, Bir, Liv}	476

This table illustrates the process of adding edges to form minimum spanning tree and calculate the total distance

Q3)

e) Prim's Algorithm for Minimum Spanning Tree (MST)

Prim's algorithm starts with an arbitary vertex and grows the minimum spanning tree (MST) by adding the smallest edge that connects a vertex in the tree to a vertex outside the tree. The process continues until all vertices are included in the MST.

1. Start with the vertex Lon:
- Choose the smallest edge connected to Lon: (Man, Yor) 80
- Now, the vertices in the MST are {Lon, Man, Yor}

2. Choose the next smallest edge connected to the current MST:
- (Oxf, Lei) 80
- Now, the vertices in the MST are {Lon, Man, Yor, Oxf, Lei}

3. Choose the next smallest edge connected to current MST:
- (Oxf, Lei) 80
- Now the vertices in MST are {Lon, Man, Yor, Oxf, Lei, Bat}

4. Choose next smallest edge connected to current MST
- ~~(Lei, Liv)~~ (Liv, Oxf) 170
- Now vertices in the MST are {Lon, Man, Yor, Oxf, Lei, Bat, Liv}

5. Choose next smallest edge connected to MST
- (Lei, Liv) 118
- Now vertices in MST are {Lon, Man, Yor, Oxf, Lei, Bat, Liv, Bir}

6. Total ditance is Minimum Spanning Tree:

$$80 + 80 + 80 + 170 + 118 = 528 \text{ miles}$$

Step	Vertex Added	Smallest Edge	MST vertices	Total Distance
1	Lon	(Man, Yor) 80	{Lon, Man, Yor}	80
2	Oxf	(Oxf, Lei) 80	{Lon, Man, Yor, Oxf, Lei}	160
3	Bat	(Oxf, Bat) 80	{Lon, Man, Yor, Oxf, Lei, Bat}	240
4	Liv	(Liv, Oxf) 170	{Lon, Man, Yor, Oxf, Lei, Bat, Liv}	410
5	Bir	(Lei, Liv) 118	{Lon, Man, Yor, Oxf, Lei, Bat, Liv, Bir}	528

Q3c)

```python
#3c
import numpy as np

# Vertices
vertices = ['Lon', 'Lei', 'Yor', 'Man', 'Liv', 'Bir', 'Oxf', 'Bat']

# Edges and weights
edges_weights = {
    ('Lon', 'Lei'): 106,
    ('Lon', 'Yor'): 211,
    ('Lon', 'Bir'): 118,
    ('Lon', 'Bat'): 118,
    ('Liv', 'Oxf'): 170,
    ('Yor', 'Bir'): 148,
    ('Man', 'Yor'): 80,
    ('Lei', 'Liv'): 118,
    ('Oxf', 'Lei'): 80,
    ('Oxf', 'Bat'): 80,
}

# Create an adjacency matrix
adjacency_matrix = np.zeros((len(vertices), len(vertices)))

for (v1, v2), weight in edges_weights.items():
    v1_index = vertices.index(v1)
    v2_index = vertices.index(v2)
    adjacency_matrix[v1_index, v2_index] = weight

# Calculate the matrix to the power of 3
matrix_power_3 = np.linalg.matrix_power(adjacency_matrix, 3)

# Find the maximum value in the matrix
```

```python
    v2_index = vertices.index(v2)
    adjacency_matrix[v1_index, v2_index] = weight

# Calculate the matrix to the power of 3
matrix_power_3 = np.linalg.matrix_power(adjacency_matrix, 3)

# Find the maximum value in the matrix
max_walks = int(matrix_power_3.max())

# Find the indices (i, j) where the maximum value occurs
indices = np.where(matrix_power_3 == max_walks)

# Extract the corresponding cities
city1_index, city2_index = indices[0][0], indices[1][0]
city1, city2 = vertices[city1_index], vertices[city2_index]

print(f"The greatest number of walks of length 3 is {max_walks} between {city1} and {city2}.")

# Display two walks of length 3 between the two cities
walk1 = f"{city1} -> {vertices[np.argmax(adjacency_matrix[city1_index])]}" \
        f" -> {vertices[np.argmax(adjacency_matrix[np.argmax(adjacency_matrix[city1_index])])]} -> {city2}"

walk2 = f"{city2} -> {vertices[np.argmax(adjacency_matrix[city2_index])]}" \
        f" -> {vertices[np.argmax(adjacency_matrix[np.argmax(adjacency_matrix[city2_index])])]} -> {city1}"

print(f"\nTwo walks of length 3 between {city1} and {city2}:\n1. {walk1}\n2. {walk2}")
```

```
The greatest number of walks of length 3 is 2126360 between Lon and Oxf.

Two walks of length 3 between Lon and Oxf:
1. Lon -> Yor -> Bir -> Oxf
2. Oxf -> Lei -> Liv -> Lon
```

```python
# Find two cities with no walks of length 3 between them
no_walks_cities = []

for i in range(len(vertices)):
    for j in range(i + 1, len(vertices)):
        if matrix_power_3[i, j] == 0 and matrix_power_3[j, i] == 0:
            no_walks_cities.append((vertices[i], vertices[j]))

print(f"\nTwo cities with NO walks of length 3 between them:")
for city_pair in no_walks_cities:
    print(f"{city_pair[0]} and {city_pair[1]}")
```

```
Two cities with NO walks of length 3 between them:
Lon and Lei
Lon and Yor
Lon and Man
Lon and Liv
Lon and Bir
Lon and Bat
Lei and Yor
Lei and Man
Lei and Liv
Lei and Bir
Lei and Oxf
Yor and Man
Yor and Liv
Yor and Bir
Yor and Oxf
Yor and Bat
Man and Liv
Man and Bir
Man and Oxf
```

```
Man and Bir
Man and Oxf
Man and Bat
Liv and Bir
Liv and Oxf
Liv and Bat
Bir and Oxf
Bir and Bat
Oxf and Bat
```

```python
#3c

import numpy as np


# Vertices
```

```python
vertices = ['Lon', 'Lei', 'Yor', 'Man', 'Liv', 'Bir', 'Oxf', 'Bat']

# Edges and weights
edges_weights = {
    ('Lon', 'Lei'): 106,
    ('Lon', 'Yor'): 211,
    ('Lon', 'Bir'): 118,
    ('Lon', 'Bat'): 118,
    ('Liv', 'Oxf'): 170,
    ('Yor', 'Bir'): 148,
    ('Man', 'Yor'): 80,
    ('Lei', 'Liv'): 118,
    ('Oxf', 'Lei'): 80,
    ('Oxf', 'Bat'): 80,
}

# Create an adjacency matrix
adjacency_matrix = np.zeros((len(vertices), len(vertices)))

for (v1, v2), weight in edges_weights.items():
    v1_index = vertices.index(v1)
    v2_index = vertices.index(v2)
    adjacency_matrix[v1_index, v2_index] = weight

# Calculate the matrix to the power of 3
matrix_power_3 = np.linalg.matrix_power(adjacency_matrix, 3)

# Find the maximum value in the matrix
max_walks = int(matrix_power_3.max())
```

```
# Find the indices (i, j) where the maximum value occurs

indices = np.where(matrix_power_3 == max_walks)


# Extract the corresponding cities

city1_index, city2_index = indices[0][0], indices[1][0]

city1, city2 = vertices[city1_index], vertices[city2_index]


print(f"The greatest number of walks of length 3 is {max_walks} between {city1}
and {city2}.")


# Display two walks of length 3 between the two cities

walk1 = f"{city1} -> {vertices[np.argmax(adjacency_matrix[city1_index])]}" \

        f" ->
{vertices[np.argmax(adjacency_matrix[np.argmax(adjacency_matrix[city1_index])])]}
-> {city2}"


walk2 = f"{city2} -> {vertices[np.argmax(adjacency_matrix[city2_index])]}" \

        f" ->
{vertices[np.argmax(adjacency_matrix[np.argmax(adjacency_matrix[city2_index])])]}
-> {city1}"


print(f"\nTwo walks of length 3 between {city1} and {city2}:\n1. {walk1}\n2.
{walk2}")


# Find two cities with no walks of length 3 between them

no_walks_cities = []


for i in range(len(vertices)):

    for j in range(i + 1, len(vertices)):

        if matrix_power_3[i, j] == 0 and matrix_power_3[j, i] == 0:
```

```
            no_walks_cities.append((vertices[i], vertices[j]))

print(f"\nTwo cities with NO walks of length 3 between them:")
for city_pair in no_walks_cities:
    print(f"{city_pair[0]} and {city_pair[1]}")
```

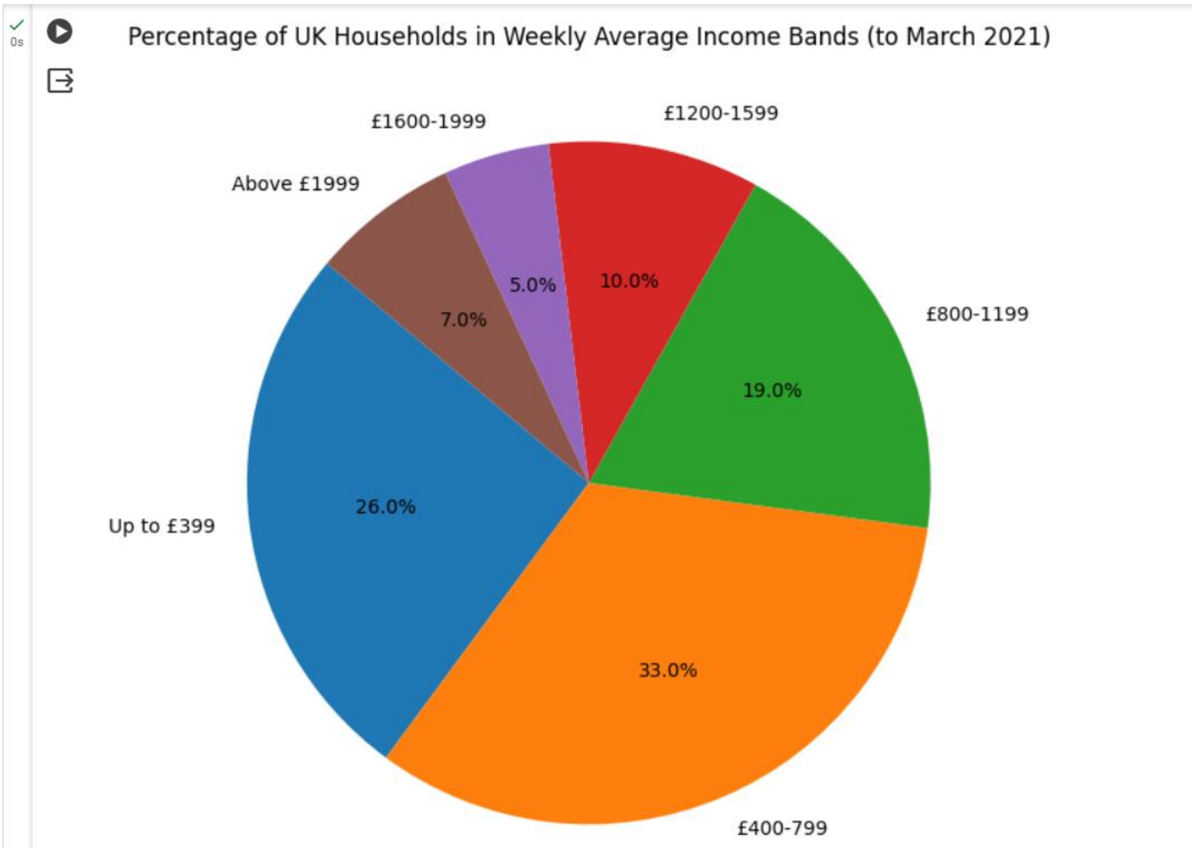
Q4

```
import matplotlib.pyplot as plt

# Data
income_bands = ['Up to £399', '£400-799', '£800-1199', '£1200-1599', '£1600-1999', 'Above £1999']
percentages = [26, 33, 19, 10, 5, 7]

# Plotting the pie chart
plt.figure(figsize=(8, 8))
plt.pie(percentages, labels=income_bands, autopct='%1.1f%%', startangle=140)
plt.title('Percentage of UK Households in Weekly Average Income Bands (to March 2021)')

# Display the pie chart
plt.show()
```



ANSWER

```python
import matplotlib.pyplot as plt


# Data

income_bands = ['Up to £399', '£400-799', '£800-1199', '£1200-1599', '£1600-1999', 'Above £1999']

percentages = [26, 33, 19, 10, 5, 7]


# Plotting the pie chart

plt.figure(figsize=(8, 8))

plt.pie(percentages, labels=income_bands, autopct='%1.1f%%', startangle=140)

plt.title('Percentage of UK Households in Weekly Average Income Bands (to March 2021)')
```

```
# Display the pie chart

plt.show()
```

Q 5)

Q5)

The sample mean $(\bar{x})$ is given by the formula

$$\bar{x} = \frac{\sum_{i=1}^{n} x_1}{n}$$

The sample variance (s^2) is given by formula.

$$s^2 = \frac{\sum_{i=1}^{n} (x_1 - \bar{x})^2}{n-1}$$

Given that $\bar{x} = 98$ and $s^2 = 120$ and the dataset $(78, 98, 102, 110, x, y)$, we can use the provided values to set up equations to solve for the unknown values x and y.

Calculate the sum of dataset and sum of squared deviation

$$\text{sum} = 78 + 98 + 102 + 110 + x + y$$

$$\text{Sum of Squared Deviations} = (78-98)^2 + (98-98)^2 + (102-98)^2 + (110-98)^2 + (x-98)^2 + (y-98)^2$$

$$\bar{x} = \frac{\text{Sum}}{6}$$

$$s^2 = \frac{\text{Sum of Squared Deviation}}{5}$$

Substitute given values:

$$98 = \frac{\text{Sum}}{6}$$

$$120 = \frac{\text{Sum of Squared Deviations}}{5}$$

1) Now solve for Sum $= 6 \times 98 = 588$

2) Solve for the sum of squared deviations:

$$120 \times 5 = \text{Sum of Squared Deviations.}$$

$$600 = (78-98)^2 + (98-98)^2 + (102-98)^2 + (110-98)^2 + (110-98)^2 + (x-98)^2 + (y-98)^2$$

Now, substitute the known values and solve for x and y

$$600 = 20^2 + 0^2 + 4^2 + 12^2 + (x-98)^2 + (y-98)^2$$

$$600 = 400 + 16 + 144 + (x-98)^2 + (y-98)^2$$

$$600 = 500 + (x-98)^2 + (y-98)^2$$

$$40 = (x-98)^2 + (y-98)^2$$

Now, consider two unknown values x and y.
You need to find values for x and y that satisfy both the equation for the sum and the equation for the sum of squared deviations. One possibility is $x = 92$ and $y = 104$ ~~but there may be others~~

V~~[illegible]~~

Q 6)

Q6)

(a) The first person called to board the plane is NOT a child

$$P(\text{Not } B) = P(\text{Adult}) = P(A \cap \text{Adult}) + P(\text{Female} \cap \text{Adult})$$

$$P(\text{Not } B) = \frac{50}{100 + 10 \times 3} + \frac{20}{30 + 10 \times 3}$$

$$P(\text{Not } B) \approx \frac{50}{130} + \frac{20}{60}$$

$$P(\text{Not } B) \approx \frac{50}{130} + \frac{1}{3}$$

$$P(\text{Not } B) \approx 0.38$$

(b) The first TWO people called to board the plane are female and the next TWO are male:-

$$P(\text{Female, Female, Male, Male}) = P(\text{Female}) \times P(\text{Female}) \times P(\text{Male}) \times P(\text{Male})$$

$$P(\text{Female, Female, Male, Male}) = \frac{30 + 10 \times 3}{100 + 10 \times 3} \times \frac{30 + 10 \times 3 - 1}{100 + 10 \times 3 - 1} \times$$

$$\frac{50}{100 + 10 \times 3 - 2} \times \frac{50 - 1}{100 + 10 + 3 - 3}$$

$$P(\text{female, Female, Male, Male}) \approx \frac{40}{130} \times \frac{39}{129} \times \frac{50}{128} \times \frac{49}{127}$$

$$P(\text{Female, Female, Male, Male}) \approx 0.18$$

(C) The first person called to board the plane is either a male, a child or both:

$$P(A \cup B) = P(A) + P(B) - P(A \cap B)$$

$$P(A \cup B) = \frac{50}{100 + 10 \times 3} + \frac{30 + 10 \times 3}{100 + 10 \times 3} - \frac{20}{100 + 10 \times 3}$$

$$P(A \cup B) \approx \frac{50}{130} + \frac{1}{3} - \frac{20}{130}$$

$$P(A \cup B) \approx \frac{30}{130} + \frac{1}{3}$$

$$P(A \cup B) \approx 0.35$$