Problem 1:

Using Python, implement the MP Neuron model taught in class that takes binary inputs (which can be both excitatory and inhibitory) and produces a binary output. For this problem, you can use the following class structure to define MP neurons:

```
class MPNeuron:
    def __init__(self, threshold, input_types):
        self.threshold = threshold
        self.input_types = input_types  # specify if each input is
excitatory or inhibitory

    def compute_output(self, inputs):
        # Implement the MP Neuron model here

        # Return the output of the MP Neuron model
        pass
```

Test your implementation with different inputs and thresholds to verify its correctness. You may use 1 and 0 to represent the two boolean logic levels (high and low).

Using the implementation of the MP Neuron discussed above, perform the following:

1) Implement the AND logic gate for two inputs [1 Mark]
2) Implement the OR logic gate for two inputs [1 Mark]
3) Implement the NOT logic gate [1 Mark]
4) Implement the NOR logic gate for two inputs [2 Marks]
5) Implement the boolean function for a scenario where a staircase light is controlled by two switches, one at the top of the stairs and other at the bottom of stairs, such that light is ON when and only when one of the switches is ON and other switch is OFF. [2 Marks]
6) What is the logic gate corresponding to the boolean function you implemented in part 5? Can you now design a multi-input version of the same logic gate in a computationally efficient manner, i.e. using as few neurons as possible? You can demonstrate the output assuming that the gate takes four inputs. [3 Marks]

Problem 2:

As taught in the class, Principal Component Analysis (PCA) can be a great tool in dimensionality reduction and compression. In this question you are required to do image compression using PCA implemented from scratch. You will be using the MNIST dataset for the same. (Using library implementations of PCA is not allowed). [15 Marks]

1) Download the dataset and visualise the digits (0-9). Report the shape of a single image. (Note: You can use `tensorflow.keras.datasets` to download the MNIST dataset) [1 Mark]

2) Randomly choose 10000 images from the dataset and flatten each image into a single NumPy array. Create a matrix X by stacking all the flattened images of the chosen digit such that each row represents a single image. [2 Marks]
3) Implement a function *standardise (X)* which takes a NumPy 2D array as an input and returns the standardised version Z. [2 Marks]
4) Standardise the X and find out the projection matrix P consisting of principal components in a computationally efficient way. [3 Marks]
5) Vary the number of selected principal components (p) from 24 to 784 (both inclusive) with the gap of 20 and reduce the dimension of the Z using the selected principal components. For each p, reconstruct the X back and visualise the first row/image in the reconstructed X. Calculate the reconstruction error of the same using L2 norm. [5 Marks]
6) Plot the reconstruction error vs number of principal components graph and give your observations. [2 Marks]