Implement a Back-Propagation Software System. Your system should permit specification of:

1. number of layers
2. number of neurons in each layer
3. learning rate
4. training, testing, and validation files (see below)
5. number of training epochs

Inputs and outputs to the net are floating-point numbers.

Your program takes three files as input (it is permissible to combine them into one file divided into three parts):

**Training Set** — used for training the net by modifying the weights according to the back-propagation algorithm.

**Testing Set** — used at the end of each epoch to test generalization (the weights are not modified).

**Validation Set** — used at the end of training to evaluate the performance of the trained net.
You will generate two sets of Training/Testing/Validating data from the following functions:

**Problem 1:**

\[ f(x, y) = \sin(\pi x) \cos(\pi y), \quad x \in (-2, 2), y \in (-2, 2). \]

**Problem 2:**

\[ f(x, y, z) = \frac{x^2}{2} + \frac{y^2}{3} + \frac{z^2}{4}, \quad x \in (-2, 2), y \in (-2, 2), z \in (-2, 2). \]

In each case generate (input, output) pairs from random inputs in the ranges specified, and outputs determined by the above formulas. For each problem, generate:

- 200 training patterns
- 100 testing patterns
- 50 validation problems

For each Problem, do the following experiments:

- Experiment with the number of hidden layers.
- Experiment with the number of neurons in each hidden layer.
- Try to determine the optimum network architecture for each problem.
- Try to determine how sensitive the performance is to the architecture.

In evaluating the architectures, pay attention to performance on the training, testing, and validation datasets.

**For graduate credit:** In addition to the preceding, explore your network’s ability to classify points generated from the following two overlapping two-dimensional Gaussians:

\[
A(x, y) = \frac{1}{2\pi} \exp \left( -\frac{x^2 + y^2}{2} \right)
\]

\[
B(x, y) = \frac{1}{8\pi} \exp \left( -\frac{(x - 2)^2 + y^2}{8} \right)
\]

Let \( x \in (-4, 10) \) and \( y \in (-6, 6) \). Generate input-output pairs \(((x, y), +1)\) with probability \( A(x, y) \) and pairs \(((x, y), -1)\) with probability \( B(x, y) \), with equal probability for \( A \) and \( B \). Explore your network’s ability to discriminate points generated by distribution \( A \) (+1 output) from those generated by distribution \( B \) (-1 output). Experiment with network architecture, generalization, etc.