# Weekly Lab Assignment – I

**Course Code:** CS16201/CS18223/CS22203

1. Design a simplified rule-based system in the context of a smart home thermostat.
   Goal: To create a system that automatically adjusts the temperature based on specific rules.
   Sample Rules:
   **Rule 1:** If the current time is between 6:00 AM and 8:00 AM and the external temperature is below 65°F, then set the thermostat to 70°F.
   **Rule 2:** If the current time is between 8:00 AM and 5:00 PM and the external temperature is above 75°F, then set the thermostat to 72°F.
   **Rule 3:** If the current time is between 5:00 PM and 6:00 AM and the external temperature is below 60°F, then set the thermostat to 65°F.
   **Rule 4:** If the user is at home and prefers a warmer temperature, then set the thermostat to the user's preferred temperature.
   **Rule 5:** If the user is away from home, then set the thermostat to an energy-saving temperature (e.g., 60°F in winter, 80°F in summer).
2. Design a simplified rule-based system in the context of a smart home lighting.
   **Goal:** To create a system that automatically adjusts the light of the house based on specific rules.
   Sample Rules:
   **Rule 1:** Time of the day (Extract the current system time), dim lights during daytime and bright lights at night.
   **Rule 2:** Number of people inside the home. If no one is at home, switch off the lights.
   **Rule 3:** If it is past 11:00 PM, switch off the lights and switch it back on at 6:00 am in dim mode.
3. Write a Python function that simulates a simple MP Neuron. The function should take a list of non-binary inputs (age, salary, number of family members) and predict whether he/she will buy the house or not (binary output). It should return 1 if the sum of the inputs is greater than the threshold, and 0 otherwise. Generate 500 records (synthetic data) and use 70% of dataset to train the MP-Neuron model. Compute the test accuracy of the model on the test (30%) dataset.
4. Modify the basic MP Neuron model and design a Perceptron model to include weights for each input. The neuron should now take an additional argument - a list of weights and compute the weighted sum of the inputs. Determine the output based on whether this weighted sum exceeds the threshold.
5. Create a simple perceptron model to perform binary classification. The model should take two inputs, apply weights, add a bias (as the threshold), and use a step function as the activation function. Test this perceptron on a simple logic gate (AND, OR).

6. Manually adjust the weights and bias of a perceptron to model the XOR logic gate. Discuss why a single-layer perceptron cannot solve the XOR problem effectively. Implement the perceptron learning rule. Use this rule to train a perceptron on a simple dataset (e.g., a linearly separable dataset). Plot the decision boundary before and after training.
7. Investigate the impact of feature scaling and normalization on perceptron training. Use a dataset with varying scales and demonstrate how normalization affects the convergence of the perceptron.