1. Explore the Hopfield network code.
   - setup pics: initializes the network and defines the training patterns
   - setup lines: alternative patterns that are more abstract and all mutually orthogonal
   - learn(k): updates network weights by Hebbian learning on training pattern k
   - start(k, clean): starts the network in a state defined by a noisy version of training pattern k; clean ∈ [-1, 1] determines the amount of noise, with clean=1 for a pure pattern, 0<clean<1 for a noisy pattern, clean=0 for pure noise, and clean=-1 for the antipattern
   - run(steps): asynchronous updating on steps randomly chosen nodes (with replacement)
   - cycle: updates all nodes simultaneously (synchronous updating); much faster than run

(a) Train the network on the training patterns. Start it in a noisy version of a training pattern (not 3 or 9) and run it until convergence. What happens, and what does this tell you about attractors in this kind of network?

(b) Start the network in pattern 3 or 9 and run to convergence. What's happening? Do you think something like this could happen with the patterns in setup lines.m?

(c) Make sure you understand the code in learn.m and run.m, or write any questions you have.

2. Consider a Hopfield network with n units, trained by Hebbian learning on a single pattern $a^1$. That is, $a^1_i \in \{-1,1\}$ for all $i$, and the weight between any two distinct nodes is $w_{ij} = \frac{1}{n} a^1_i a^1_j$, with $w_{ii} = 0$. Prove that $a^1$ is a stable state (i.e., an attractor) of the network.

   More specifically: Imagine we put the network in state $a^1$, by setting the activation $a_j = a^1_j$ for all $j$, and we pick any node $i$ and update it according to $a_i \leftarrow \text{sign}(\sum_j a_j w_{ij})$. Prove that $a_i$ doesn't change, i.e. that $\text{sign}(\sum_j a_j w_{ij}) = a^1_i$. Hint: substitute the definition of $w$ into the update equation, and use the fact that $a_j a_j = 1$ for all $j$.

3. Now imagine training the network on two patterns, $a^1$ and $a^2$, so that $w_{ij} = \frac{1}{n} a^1_i a^1_j + \frac{1}{n} a^2_i a^2_j$ for all $i \neq j$ and $w_{ii} = 0$.

   (a) Assume the network is in state $a^1$. Write an expression for the total input to any node $i$, in terms of $a^1$ and $a^2$ (i.e., eliminating $w$). Simplify the expression as much as possible, to separate the interference between $a^1$ and $a^2$ from the contribution of $a^1$ alone (the latter should match what you derived in question 2).

   (b) Comparing the two terms in the previous answer (interference and contribution from $a^1$ alone), try to figure out what would need to happen for the training patterns not to be stable. That is, how would $a^1$ and $a^2$ need to be related in order for the interference terms to cause a problem?