THE FEEDFORWARD NETWORK!

SETUP:

- 5 layers
- N neurons per layer
- Each neuron either spiking (1) or quiet (0)
- State of each layer is list of N 0's and 1's: the vectors `x_1`, `x_2`, ... `x_5`

DYNAMICS:

- Each neuron gets an input from previous layer. If this input is above its threshold, it spikes. If the input is below this threshold, it is quiet.
  - We will set the threshold for all neurons to 1

The input to each neuron is determined as follows:

1. Each neuron that spiked in the previous layer attempts to communicate to every neuron in the current layer. This communication is in terms of synaptic events.
2. This communication succeeds with probability (frequency) `p`. Otherwise, it fails.
3. The input to a cell is the total number of synaptic events that it successfully receives

CODES:

`network_run.m`

- Produces the `x_1`, `x_2`, ... `x_5` for **ONE** run ("realization") of the network dynamics

`network_iterate.m`

- sets the parameter N
- Runs the program `network_run.m` many times
- Analyzes and plots the results.

NEXT STEPS:

TEAM 1:

- Modify `network_iterate.m` so that it loops over 50 trials. It should build a table `n_table` where every row is `n_list` for one trial. Hint: Make a large matrix of 0's, and then fill in one row at a time.
- Modify `network_iterate.m` so that on each trial there is a new random vector `x_1` of firing events in the first layer. This should be a vector of 0's and 1's, with "1" occurring with probability `r_1` (the firing rate for the first layer)
- Define the probability that a cell is spiking in the final layer (layer 5) as `r_out`. Come up with a formula for this using existing variables, and compute it in your code.

TEAM 2:

- Modify `network_run` so that (see HINT below, with nested for loops):
  - There is a P matrix (e.g., `P_12`) for each W matrix (e.g. `W_12`).
  - Entries of each W matrix are 0's or 1's. Each entry (i.e., `W_12(i,j)`) should be 1 with probability given by the corresponding P matrix entry (e.g., `P_12(i,j)`).

CULMINATION:

Now, merge codes from teams 1 and 2 ...

HINT: This code modifies network to make synapses probabilistic with success proba 1/N

```
for j = 1:N
    for k = 1:N
        if rand>1/N
            W_12(j,k)=0;
        end
        if rand>1/N
            W_23(j,k)=0;
        end
        if rand>1/N
            W_34(j,k)=0;
        end
        if rand>1/N
            W_45(j,k)=0;
        end
    end
end
```

Next ...

OUR QUESTION: How well does network transmit information?

- Imagine light vs dark encoded by firing rates r_{in}, in layer 1.
- How is the range of input rates mapped into output rates r_{out}

d ...

TEAM 1:

- Modify the latest `network_iterate.m` to loop over 10 values of `r_in`, via a "big" outer for loop.
- Add command to plot `r_out` vs. `r_in`. This characterizes the signal transmission performance of your network!
- Use subplot command to plot the average number of active cells per layer vs. layer index, for each value of r_in.

TEAM 2: Meanwhile, assume you're given list of r_{in} and r_{out} values

- Assess network performance via a grade (0-100)
- Say if `r_out` = `r_in` ... identity line ... then signal transmission is "optimal."
- Define d_j = distance of each (r_{in}, r_{out}) pair from identity line. Define $dmax_j$ worst case distance for each r_{in}
- Thus
$$\frac{\sum_j d_j}{\sum_j dmax_j} \tag{1}$$
is 0 for optimal transmission, and 1 for worst transmission
- Assign grade as
$$100 * \left(1 - \frac{\sum_j d_j}{\sum_j dmax_j}\right)$$