Consider the idea that all we have are different layers with activations $a_{\alpha\beta}$, for some layer α and specific activation β. The error function (aka cost function) would be:

$$E=\frac{1}{2}\sum_{I}\omega_I^2 \text{ with } \omega_I=\left(T_I-F_I\right),$$

where F_i represents the activations of the output layer. Now consider a network with five or more activation layers. I have removed the special symbols for the input and output activations that have been used in the previous documentation. The various calculations that were derived for the three activation-layer network can be structured to more clearly show the interdependencies between the layers. I am still using the same convention as in the previous documents, that the index letter is associated with a specific layer, although I am generalizing the notion in the text. You are free to use either convention.

$$a_x \quad \Theta_m=\sum_X a_X w_{Xm} \quad a_m=f\left(\Theta_m\right) \quad \Theta_k=\sum_M a_M w_{Mk} \quad a_k=f\left(\Theta_k\right) \quad \Theta_j=\sum_K a_K w_{Kj} \quad a_j=f\left(\Theta_j\right) \quad \Theta_i=\sum_J a_J w_{Ji}$$

$$a_i=f\left(\Theta_i\right)$$

$$\omega_i=\left(T_i-a_i\right)$$

$$\psi_i=\omega_i f'\left(\Theta_i\right)$$

$$\Omega_j=\sum_I \psi_I w_{jI}$$

$$w_{ji} \mathrel{+}= \lambda a_j \psi_i$$

$$\Psi_j=\Omega_j f'\left(\Theta_j\right)$$

$$\Omega_k=\sum_J \Psi_J w_{kJ}$$

$$w_{kj} \mathrel{+}= \lambda a_k \Psi_j$$

$$\Psi_k=\Omega_k f'\left(\Theta_k\right)$$

$$\Omega_m=\sum_K \Psi_K w_{mK}$$

$$w_{mk} \mathrel{+}= \lambda a_m \Psi_k$$

$$\Psi_m=\Omega_m f'\left(\Theta_m\right)$$

$$w_{xm} \mathrel{+}= \lambda a_x \Psi_m$$

You can now more clearly see the common elements in each calculation layer. Notice that the last step in each direction is slightly different and so in each case that entire loop should be broken out from the main loop construct and done as a final step.

Based on the diagram an implementation for the forward propagation is therefore:

```
for α = 2 to N_layers                // Activation layers ...,k,j,i
   for β = 1 to N_α                  // values in current layer α
      Θ_αβ = 0                       // Zero the accumulator
      for γ = 1 to N_α-1             // Values in previous layer α-1
         Θ_αβ += a_(α-1)γ w_(α-1)γβ  // weight layer n = α-1
      next γ
      a_αβ = f(Θ_αβ)
   next β
next α
```

Where N_{layers} is the number of activation layers. When these loops have completed we should have the activations. The abstraction in our mathematical notation should be

$$\boxed{a_{\alpha\beta} = f\left(\Theta_{\alpha\beta}\right) \text{ where } \Theta_{\alpha\beta} = \sum_{\gamma} a_{(\alpha-1)\gamma} w_{(\alpha-1)\gamma\beta}}$$

where α is the activation layer indexed over β and the index γ is for layer α-1, remembering that the layer indexing goes from left to right with the left most activations being the input layer (and so are not calculated, but are set as input to the network).

Now we need to rewrite the back propagation relationships in the final more generalized notation. We are going to use a single ψ array to handle what gets passed up the layers so we can simplify the back propagation loops, so we need to do an initial population for the output later. That means we need to break the forward calculations into two segments, where the output layer is handled by itself.

```
for α = 2 to N_layers-1              // layers ...,k,j, but not i
   for β = 1 to N_α                  // values in current layer α
      Θ_αβ = 0
      for γ = 1 to N_α-1             // Values in previous layer α-1
         Θ_αβ += a_(α-1)γ w_(α-1)γβ  // weight layer n = α-1
      next γ
      a_αβ = f(Θ_αβ)
   next β
next α

α = N_layers                         // Now Just the output layer i
for β = 1 to N_α                     // values in current layer α
   Θ_αβ = 0
   for γ = 1 to N_α-1                // Values in previous layer α-1
      Θ_αβ += a_(α-1)γ w_(α-1)γβ     // weight layer n = α-1
   next γ
   a_αβ = f(Θ_αβ)
   ψ_αβ = (T_β - a_αβ) f'(Θ_αβ)      // T_β is the expected output
next β
```

Notice how the last set of loop constructs is nearly identical to the previous set. All we did was fix the value of α and add the ψ calculation. Now we have the data we need to go backward through the network and train. We are going to move in the reverse direction so we start in the second to last layer, which pulls information from the output layer. We only use the Ω values in the current layer, so we just need a simple scalar value there. We need to break out the last loop because we don't really need to store any more Ψ values beyond that point and we do need to update the weights connected to the input layer.

I specifically want you to put the calculations for $\psi_{\alpha\beta} = (T_\beta - a_{\alpha\beta}) f'(\Theta_{\alpha\beta})$ in the separate final loop as shown above when doing the forward calculation and not in the back propagation. This design is my preference because it keeps the symmetry of using ψ values calculated in the layer to the right of the current layer in the back propagation logic and reflects the geometry seen in the flow diagram.

Write up your pseudocode and let me look it over. It will save you time if there are issues.

After you have submitted this project, change the threshold function to $f(x)=\frac{e^x-e^{-x}}{e^x+e^{-x}}$, which is the hyperbolic tangent and has a somewhat different behavior than a sigmoid in that it has a range from -1 to +1 and has $f'(x)=1-f^2(x)$. Try it out and see which one works better. There are issues when computing this function when $|x|$ gets large which we will discuss in class.