Problem 1

c is the correct answer.
When $N = 25$, the expected E_{in} is .0064. When $N = 100$, the expected E_{in} is .0091. Thus we have our answer.

Problem 2

d is the correct answer.
We are looking to find the correct behavior for $\text{sign}(\tilde{w}_0 + \tilde{w}_1 x_1^2 + \tilde{w}_2 x_2^2)$. We have that if $\tilde{w}_1 < 0$ and $\tilde{w}_2 > 0$, then we get a positive sign for large absolute values of x_2 and a negative sign for large absolute values of x_1. And we can adjust $\tilde{w}_0$ to achieve the exact behavior we are looking for. Thus we have our answer.

Problem 3

c is the correct answer.
We have that the VC dimension of a linear model in the transformed space follows the inequality $d_{VC} \leq \tilde{d} + 1$. And we have that $\tilde{d} = 14$, which means that $d_{VC} \leq 15$. Thus we have our answer.

Problem 4

e is the correct answer.
We just use the chain rule to take the partial derivative of $E(u, v)$ with respect to u to get the answer.

Problem 5

d is the correct answer.
I wrote a program to run gradient descent with the problem's parameters plugged in and it took 10 iterations to get the error below 10^{-14} for the first time. A basic outline of the program is the following. For each iteration, I computed the gradient of $E(u, v)$. I then used that gradient and the learning rate to update the weights, which in this case are just u and v. I then checked the error. If it was still too high, I ran another iteration.

Problem 6

e is the correct answer.
I wrote a program to run gradient descent with the problem's parameters plugged in and I ended up with $(.0447, .024)$ as the final (u, v) after the error just dropped below 10^{-14}. The program is the same program I outlined in Problem 5.

Problem 7

a is the correct answer.
I wrote a program to run coordinate descent with the problem's parameters plugged in and I ended up with an error of .14 after 15 full iterations (30 steps). A basic outline of the program is the following. For each iteration, I computed the gradient, then updated u. Then, I recomputed the gradient, and updated v. I did ran this iteration 15 times.

Problem 8

a is the correct answer.
I wrote a program to run Stochastic Gradient Descent on 100 random training points to find a hypothesis g, which in this case is a weight vector $\mathbf{w}$. After running SGD to get $\mathbf{w}$, I calculated E_{out} by generating

a large, separate set of points $\mathbf{x}_1, \cdots, \mathbf{x}_k$ and comparing the sign of $\mathbf{w}^\intercal \mathbf{x}_i$ to the sign the target function would have assigned $\mathbf{x}_i$ and seeing what fraction of the signs differed. I then repeated this experiment with different targets and sample points for 1000 runs and took the average E_{out}, which was around .021.

Problem 9

a is the correct answer.
I wrote a program to run Stochastic Gradient Descent on 100 random training points using the initialization and termination rules as specified in the problem (as well as the learning rate specified in the problem). I ran the SGD to convergence 1000 times and calculated the average number of epochs, which was around 340.

Problem 10

e is the correct answer.
In SGD, we update the weights using

$$\mathbf{w}(t+1) = \mathbf{w}(t) - \eta \nabla E_{\text{in}}$$

In PLA, we update the weight vectors using

$$\mathbf{w}(t+1) = \mathbf{w}(t) + y\mathbf{x}$$

where (x, y) is a misclassified training point. So basically, we want the gradient of $e_n(\mathbf{w})$ to be equal to 0 if $\mathbf{x}$ is not a misclassified point or $-y\mathbf{x}$ if $\mathbf{x}$ is a misclassified point. And **e** gives us that, since if $\mathbf{x}$ is a misclassified point, then $y\mathbf{w}^\intercal \mathbf{x}$ will be negative and thus less than 0. Thus we have our answer.