Problem 1

b is the correct answer.
This is because when we use a less sophisticated hypothesis set, we are going to be able, in general, to capture less of the target function f.

Problem 2

a is the correct answer.
I wrote some code to do it. More specifically, I wrote code that runs Linear Regression on the training set after performing the non-linear transformation. This basically just boils down to calculating the weight vector by multiplying a bunch of matrices, as seen in lecture 12 slide 11 (no regularization). I then calculated the in-sample and out-of-sample classification errors in the usual manner (using the weight vector I calculated, calculating dot products, and comparing signs).

Problem 3

d is the correct answer.
I wrote some code to do it. More specifically, I wrote code that adds weight decay to Linear Regression. This basically just boils down to calculating the weight vector w_{reg} by multiplying a bunch of matrices, as seen in lecture 12 slide 11. In this multiplication, I used the term given to us in the problem and $k = -3$. I then calculated the in-sample and out-of-sample classification errors in the usual manner.

Problem 4

e is the correct answer.
I wrote some code to do it. More specifically, I wrote code that adds weight decay to Linear Regression. This basically just boils down to calculating the weight vector w_{reg} by multiplying a bunch of matrices, as seen in lecture 12 slide 11. In this multiplication, I used the term given to us in the problem and $k = 3$. I then calculated the in-sample and out-of-sample classification errors in the usual manner.

Problem 5

d is the correct answer.
I wrote some code to do it. More specifically, I used the code I wrote for the previous two parts and calculated the out-of-sample classification error for each of the k values given in the choices. I got the smallest out-of-sample classification error when $k = -1$.

Problem 6

b is the correct answer.
I wrote some code to do it. More specifically, I looped through a large range of ks (positive and negative) and kept track of the minimum out-of-sample classification error. The minimum I found was .056, which is closest to .06.

Problem 7

c is the correct answer.
c is the correct answer because this intersection makes it so that when $q > 2$, $w_q = 0$ (for lower terms, the weights are free to be anything). This is because when we take the intersection we just get the set on the left because it is included in the set on the right. This is clear to see because the set on the left forces all terms where $q \geq 3$ to be zero, and the set on the right forces all terms where $q \geq 4$ to be zero. So the set on the left is clearly the intersection of both the sets, and since the set on the left is exactly what $\mathcal{H}_2$ is, **c** is our answer.

Problem 8

d is the correct answer.
For the operations including $w_{ij}^{(l)} x_i^{(l-1)}$, which are of the form

$$x_j^{(l)} = \theta(\sum_{i=0}^{d^{(l-1)}} w_{ij}^{(l)} x_i^{(l-1)})$$

I counted 22 operations (18 for $l = 1$, $1 \leq j \leq 3$ and 4 for $l = 2$, $1 \leq j \leq 1$). For the operations including $w_{ij}^{(l)} \delta_j^{(l)}$, which are of the form

$$\delta_i^{(l-1)} = (1 - (x_i^{(l-1)})^2) \sum_{j=1}^{d^{(l)}} w_{ij}^{(l)} \delta_j^{(l)}$$

I counted 3 operations (0 for $l = 1$, $1 \leq i \leq 5$ because for the input layer there is no signal and 3 for $l = 2$, $1 \leq i \leq 3$; $i = 0$ was not counted because it makes the coefficient in front of the sum be just zero). For the operations including $x_i^{(l-1)} \delta_j^{(l)}$, which are of the form

$$w_{ij}^{(l)} \leftarrow w_{ij}^{(l)} - \eta x_i^{(l-1)} \delta_j^{(l)}$$

I counted 22 operations (18 for $l = 1$, $1 \leq j \leq 3$, $1 \leq i \leq 5$ and 4 for $l = 2$, $1 \leq j \leq 1$, $0 \leq i \leq 3$). These counts give me a total of 47.

Problem 9

a is the correct answer.
Since each layer is fully connected, to minimize the number of weights, we only want to put two units on each hidden layer. Since no weights go into the $x_0^{(l)}$s, we have that this gives us 10 weights from the input units to the first hidden layer and 36 weights total from the hidden units to other hidden units/the output unit (one weight per hidden unit, as for each hidden layer, we will have one weight going from each hidden unit to the non $x_0^{(l)}$ unit of the next layer, or in the case of the last hidden layer, going to the output unit).

Problem 10

e is the correct answer.
It seemed like 2 hidden levels would maximize the number of weights. So I came up with a function for this situation,

$$|w| = 10(x - 1) + x(36 - x - 1) + (36 - x)$$

where x is the number of units in the first hidden level and $|w|$ is the number of weights in the network. This function is based off of the fact that each layer is fully connected to the next layer. I then maximized this function in Wolfram Alpha and it gave me that $|w|_{max} = 510$ when $x = 22$. And since no answers are greater than 510, this must be our answer.