Problem 1

b is the correct answer.
I got this answer by running the experiment detailed in the problem 100,000 times, and averaging the ν_{min}.

Problem 2

d is the correct answer.
In the simulation I ran, my averages for ν_1 and ν_{rand} were basically .5. And since $\mu = .5$, we have that the coins c_1 and c_{rand} satisfy the Hoeffding Inequality.

Problem 3

e is the correct answer.
The first way to get an error is when $y = f(x)$ but the hypothesis makes an error in approximating the deterministic function. The probability of this happening is $\lambda * \mu$. The second way to get an error is when $y \neq f(x)$ and the hypothesis does not make an error in approximating the deterministic function. The probability of this happening is $(1 - \lambda) * (1 - \mu)$. So overall we get that the probability of error that h makes in approximating y is $(1 - \lambda) * (1 - \mu) + \lambda * \mu$.

Problem 4

e is the correct answer.
By looking at our answer to Problem 3, which is $2 * \lambda * \mu - \lambda - \mu + 1$ which is equivalent to $(1 - \lambda) * (1 - \mu) + \lambda * \mu$, , we can see that at no values of λ will the performance of h be independent of μ (we can't get rid of μ in that equation by setting λ to any of the chooseable values).

Problem 5

c is the correct answer.
I used Linear Regression to find the weight vector for some 100 points as described in the problem, and used this weight vector to calculate the fraction of misclassified in-sample points. I then repeated this 1000 times and took the average, and **c** was the closest number to the average I got.

Problem 6

c is the correct answer.
I used Linear Regression to find the weight vector for some 100 points as described in the problem. I then generated 1000 fresh points, and used the weight vector to see what fraction of these points were misclassified. I then repeated this 1000 times and took the average, and **c** was the closest number to the average I got.

Problem 7

a is the correct answer.
I used Linear Regression to find the weight vector for some 10 points as described in the problem. I then ran the PLA on those points, using the weight vector I found as the initial weight vector in this algorithm. I then repeated this 1000 times and took the average number of iterations of the PLA, and **a** was the closest number to the average I got.

Problem 8

d is the correct answer.
I generated a training set of 1000 points, assigned them scores according to the target function given in the problem, and generated simulated noise as instructed in the problem. I then used Linear Regression to find the weight vector for this set of points, and used the weight vector to calculate the fraction of misclassified in-sample points. I then repeated this 1000 times and took the average, and **d** was the closest to the average I got.

Problem 9

a is the correct answer.
I transformed the training data into nonlinear feature vectors as described in the problem. I then used Linear Regression on the set of feature vectors to find the weight vector. I repeated this 1000 times, and took the average weight vector. **a** was the closest to the average weight vector I got. I also checked this by comparing the **a-e** hypotheses point by point (with a batch of test points) to the hypothesis I found, and **a** disagreed on the least amount of points.

Problem 10

b is the correct answer.
I did the same as in Problem 9, running Linear Regression on the set of feature vectors to find the weight vector. Then, I generated a new set of 1000 points and added noise as before. By using the output of these points, the feature vectors of these points, and the weight vector, I was able to find the fraction of the points that were misclassified. I then repeated this 1000 times and took the average, and **b** was the closest to the average I got.