Problem 1

d is the correct answer.
I wrote some code that ran linear regression on the first 25 examples on **in.dta** using each of the given models. This results in a a weight vector. I then used this weight vector to calculate the out-of-sample error, using the validation set (the last 10 examples of **in.dta**) as the out-of-sample points. The model with $k = 6$ resulted in the smallest out-of-sample error.

Problem 2

e is the correct answer.
I wrote some code that ran linear regression on the first 25 examples on **in.dta** using each of the given models. This results in a a weight vector. I then used this weight vector to calculate the out-of-sample error, using **out.dta** as the out-of-sample points. The model with $k = 7$ resulted in the smallest out-of-sample error.

Problem 3

d is the correct answer.
I wrote some code that ran linear regression on the last 10 examples on **in.dta** using each of the given models. This results in a a weight vector. I then used this weight vector to calculate the out-of-sample error, using the validation set (the first 25 examples of **in.dta**) as the out-of-sample points. The model with $k = 6$ resulted in the smallest out-of-sample error.

Problem 4

d is the correct answer.
I wrote some code that ran linear regression on the last 10 examples on **in.dta** using each of the given models. This results in a a weight vector. I then used this weight vector to calculate the out-of-sample error, using **out.dta** as the out of sample points. The model with $k = 6$ resulted in the smallest out-of-sample error.

Problem 5

b is the correct answer.
In Problem 1, I chose the model with $k = 6$, trained on the first 25 points from **in.dta**. The out-of-sample classification error for this model is .084. In Problem 3, I chose the model with $k = 6$, trained on the last 10 points from **in.dta**. The out-of-sample classification error for this model is .192.

Problem 6

d is the correct answer.
I wrote code that generated two random variables distributed uniformly over the interval $[0, 1]$. I then took the minimum of these two values. I then ran this code 1000 times and took the average of all three values. The average value for the minimum I got was around .33.

Problem 7

c is the correct answer.
I wrote code that ran leave-one-out cross-validation with the three given points for both of the two models. This code gave me the cross-validation errors. So, for each choice of ρ, I ran this code, getting the cross-validation error for both models. For choice **c**, these two values were equal.

Problem 8

c is the correct answer.
I wrote code that used PLA to find $E_{\text{PLA}} = \mathbb{P}[f(\mathbf{x}) \neq g_{\text{PLA}}(\mathbf{x})]$ and that used SVM to find $E_{\text{SVM}} = \mathbb{P}[f(\mathbf{x}) \neq g_{\text{SVM}}(\mathbf{x})]$. To run SVM, I used the python cvxopt module to solve the dual QP. This was all for $N = 10$. For each valid run, I compared the two to see if E_{SVM} was lower. I counted how many times this occured and divided it by the number of valid runs, using around 1000 valid runs. This value was around .6.

Problem 9

CORRECTION: d IS ACTUALLY THE CORRECT ANSWER
Choose two: **c** and **d**
I wrote code that used PLA to find $E_{\text{PLA}} = \mathbb{P}[f(\mathbf{x}) \neq g_{\text{PLA}}(\mathbf{x})]$ and that used SVM to find $E_{\text{SVM}} = \mathbb{P}[f(\mathbf{x}) \neq g_{\text{SVM}}(\mathbf{x})]$. To run SVM, I used the python cvxopt module to solve the dual QP. This was all for $N = 100$. For each valid run, I compared the two to see if E_{SVM} was lower. I counted how many times this occured and divided it by the number of valid runs, using around 1000 valid runs. This value was around .6 (sometimes over, usually under).

Problem 10

b is the correct answer.
I wrote code that ran SVM for 1000 runs on 100 points. To run SVM, I used the python cvxopt module to solve the dual QP. For each run, I kept track of the number of support vectors by counting the number of α_ns that were greater than 0 (actually counted the number that were greater than 10^{-5}, since basically all of them were greater than 0). I averaged these numbers out for the 1000 runs and ended up with a number around 3.