1. Deriving the batch gradient descent equations for logistic regression with l_2 regularization:
   Using the same notation in class, $\mu(x) = \frac{1}{1+\exp(-B^T x)}, \eta(x) = B^T x$, write the negative log likelihood with the regularization factor:
   $L(\beta|x) = -\sum_i (y_i * \ln(\mu_i) + (1 - y_i) * \ln(1 - \mu_i)) + c||\beta||^2$
   We need the gradient, so differentiate
   $\frac{dL}{d\beta} = -\sum_i \left(y_i \frac{1}{\mu_1} \frac{d\mu_i}{d\beta} - \frac{1-y_i}{1-\mu_i} \frac{d\mu_i}{d\beta} \right) + 2c\beta$
   $\frac{dL}{d\beta} = -\sum_i ((y_i - \mu_i)x_i) + 2c\beta$
   $\beta^{\text{new}} = \beta^{\text{old}} - \rho \nabla L$
   $\beta^{\text{new}} = \beta^{\text{old}} + \rho \sum_i ((y_i - \mu_i)x_i) - \rho 2c\beta^{\text{old}}$

   You can generate the plots by running prob1
   The errors on the training and test sets are on the plots p1_i, p1_ii, p1_iii for the preprocessing options i,ii,iii respectively. They are labeled as (training_err, test_err)

2. Now we pick a random point i and use that point as the gradient instead of summing over all of the data points, so the new update would be
   $\beta^{\text{new}} = \beta^{\text{old}} + \rho(y_i - \mu_i)x_i - 2\rho c\beta^{\text{old}}$
   It seems with these curves that we can make the loss arbitrarily small, whereas we couldn't necessarily do that with 1), I suppose this is because I don't have the computing power to get the number of iterations for 1). The error rates weren't definitively better for one versus the other.. I'd be interested in more heavily tuning the parameters using stochastic gradient descent because I believe its more promising when we have lots of training data.

   You can generate the plots by running prob2
   The errors on the training and test sets are on the plots p2_i, p2_ii, p2_iii for the preprocessing options i,ii,iii respectively. They are labeled as (training_err, test_err)

3. This strategy does ok, but using a small enough rho and large enough iterations seemed to work better. We could also conceivably set rho proportional to $\frac{1}{n^2}$, or maybe make rho a function of how much our gradient vector is changing each time. I.e. if we have big updates several times in a row, then we should have a bigger rho because our estimate is most likely off. Rho could then be an exponential moving average of the updates.

   You can generate the plot by running prob3

4. The plot can be found in p4.png or generated by running prob4
   You can find the training error and test error by running prob4_2, which I got to be

8.81% for the training set and 9.64% for the test set