```
function sim = gaussianKernel(x1, x2, sigma)
%RBFKERNEL returns a radial basis function kernel between x1 and x2
%   sim = gaussianKernel(x1, x2) returns a gaussian kernel between x1 and x2
%   and returns the value in sim

% Ensure that x1 and x2 are column vectors
x1 = x1(:); x2 = x2(:);

% You need to return the following variables correctly.
sim = 0;

% ====================== YOUR CODE HERE ======================
% Instructions: Fill in this function to return the similarity between x1
%               and x2 computed using a Gaussian kernel with bandwidth
%               sigma
%
%
sim = exp(-sum((x1 - x2) .^ 2) / (2 * sigma ^ 2));
```

```
function [C, sigma] = dataset3Params(X, y, Xval, yval)
%DATASET3PARAMS returns your choice of C and sigma for Part 3 of the exercise
%where you select the optimal (C, sigma) learning parameters to use for SVM
%with RBF kernel
%   [C, sigma] = DATASET3PARAMS(X, y, Xval, yval) returns your choice of C and
%   sigma. You should complete this function to return the optimal C and
%   sigma based on a cross-validation set.
%

% You need to return the following variables correctly.
C = 1;
sigma = 0.3;

% ====================== YOUR CODE HERE ======================
% Instructions: Fill in this function to return the optimal C and sigma
%               learning parameters found using the cross validation set.
%               You can use svmPredict to predict the labels on the cross
%               validation set. For example,
%                   predictions = svmPredict(model, Xval);
%               will return the predictions on the cross validation set.
%
%  Note: You can compute the prediction error using
%        mean(double(predictions ~= yval))
```

```
%

% As per page 7 of the instructions, we are going to look at 8
% different values for both C and sigma. As a result, we will use a
% nested for loop. Remember that Xval and yval are the X and y values
% for our cross training dataset.
testValues = [0.01, 0.03, 0.1, 1, 3, 10, 30];
for i =1:length(testValues),
    for j=1:length(testValues),

     % Use the svmTrain function to get model output for each
           % possible combination of i,j (C, sigma), Note that in
           % svmTrain we have to pass in a kernelFunciton

     testModel = svmTrain(X, y, testValues(i), @(x1, x2) gaussianKernel(x1, x2,testValues(j)));

     % Now we can call svmPredict to get the predication vectors

     testPredictions = svmPredict(testModel, Xval);

     % Now we can calculate the predication eror using the above
           % formula



      crossValPredictErrors(i, j) = mean(double(testPredictions ~= yval)
     end;
end;

[minColumnError, minColumnErrorIndex] = min(crossValPredictErrors);
[minError, minErrorIndex] = min(minColumnError);

C= testValues(minColumnErrorIndex(minErrorIndex));
sigma = testValues(minErrorIndex);

% This is a tiny bit complicated, so here is what is going on.
% crossValPredictErrors is an 8x8 matrix returned from running
% svmTrain on every possible combination of C and sigma defined in
% testValues.

% minColumnError returns the minimum error from each
% column, and minColumnErrorIndex returns where along that column that
% minimum error value can be found. minColumnError returns as a 1x8
% vector of minimum error values.

% From minColumnError we extract the global minimum as minError, and
% we find its index with minErrorIndex. Now, we have the row and
```

```
% column index for the global minimum, so we take those values and
% apply them to the testValues vector to get the best possible values
% for C and for sigma.









% ========================================================================

end
```