Sigmoid:

```
function g = sigmoid(z)
g = 1./(1+exp(-z));
end
```

costFunction:

```
function [J, grad] = costFunction(theta, X, y)
m = length(y);
grad = zeros(size(theta));
htheta=sigmoid(X*theta);
J=(-y'*log(htheta)-(1-y)'*log(1-htheta))/m;
d = size(theta,1);
for j=1:d
   grad(j) = ones(1,m)*((sigmoid(X*theta)-y).*X(:,j))/m;
end
end
```

fminunc:

```
options = optimset('GradObj', 'on', 'MaxIter', 400);
[theta, cost] = ...
        fminunc(@(t)(costFunction(t, X, y)), initial_theta, options);
[theta, J, exit_flag] = ...
        fminunc(@(t)(costFunctionReg(t, X, y, lambda)), initial_theta,
options);
```

predict:

```
function p = predict(theta, X)
m = size(X, 1); % Number of training examples
p = zeros(m, 1);
htheta_est=sigmoid(X*theta);
for i=1:m
    if htheta_est(i) >= 0.5
        p(i) = 1;
    else
        p(i) = 0;
    end
end
```

costFunctionReg:

```
function [J, grad] = costFunctionReg(theta, X, y, lambda)
% Initialize some useful values
m = length(y); % number of training examples
J = 0;
grad = zeros(size(theta));
htheta=sigmoid(X*theta);
J=(-y'*log(htheta)-(1-y)'*log(1-htheta))/m + lambda/2/m*...
    (theta'*theta-theta(1)^2);
d = size(theta,1);
    grad(1) = ones(1,m)*((sigmoid(X*theta)-y).*X(:,1))/m;
for j=2:d
    grad(j) = ones(1,m)*((sigmoid(X*theta)-y).*X(:,j))/m +
lambda/m*theta(j);
end

end
```

plotDecisionBoundary:(***)

```
function plotDecisionBoundary(theta, X, y)
%PLOTDECISIONBOUNDARY Plots the data points X and y into a new figure with
%the decision boundary defined by theta
%   PLOTDECISIONBOUNDARY(theta, X,y) plots the data points with + for the
%   positive examples and o for the negative examples. X is assumed to be
%   a either
%   1) Mx3 matrix, where the first column is an all-ones column for the
%      intercept.
%   2) MxN, N>3 matrix, where the first column is all-ones

% Plot Data
plotData(X(:,2:3), y);
hold on

if size(X, 2) <= 3
    % Only need 2 points to define a line, so choose two endpoints
    plot_x = [min(X(:,2))-2,  max(X(:,2))+2];

    % Calculate the decision boundary line
    plot_y = (-1./theta(3)).*(theta(2).*plot_x + theta(1));

    % Plot, and adjust axes for better viewing
    plot(plot_x, plot_y)

    % Legend, specific for the exercise
    legend('Admitted', 'Not admitted', 'Decision Boundary')
    axis([30, 100, 30, 100])
else
    % Here is the grid range
    u = linspace(-1, 1.5, 50);
    v = linspace(-1, 1.5, 50);
```

```
    z = zeros(length(u), length(v));
    % Evaluate z = theta*x over the grid
    for i = 1:length(u)
        for j = 1:length(v)
            z(i,j) = mapFeature(u(i), v(j))*theta;
        end
    end
    z = z'; % important to transpose z before calling contour

    % Plot z = 0
    % Notice you need to specify the range [0, 0]
    contour(u, v, z, [0, 0], 'LineWidth', 2)
end
hold off

end
```