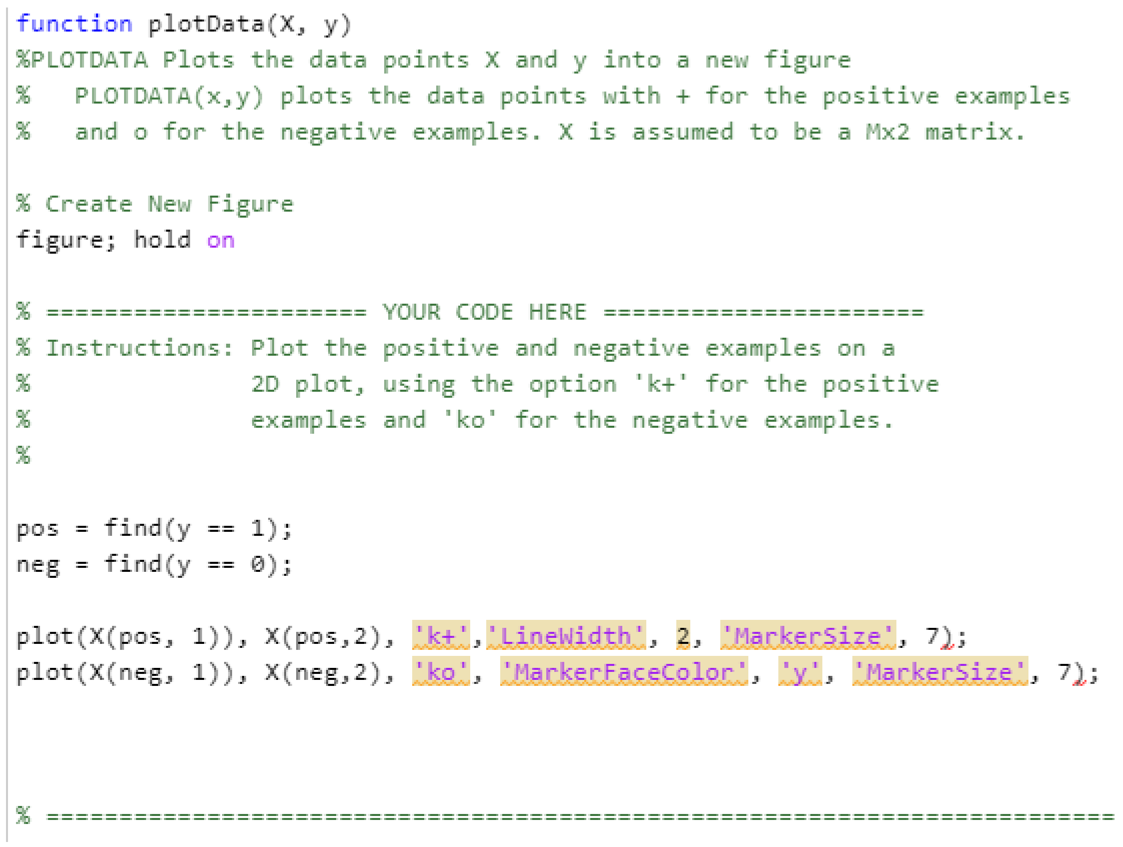
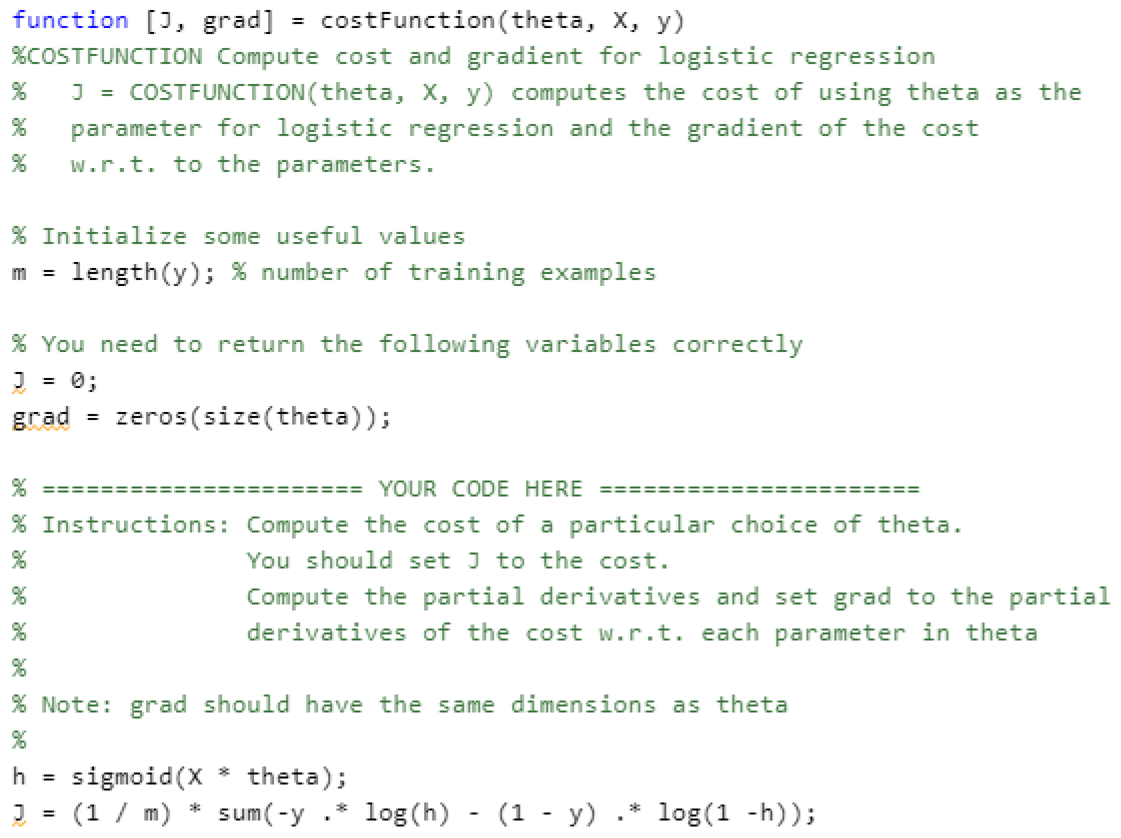


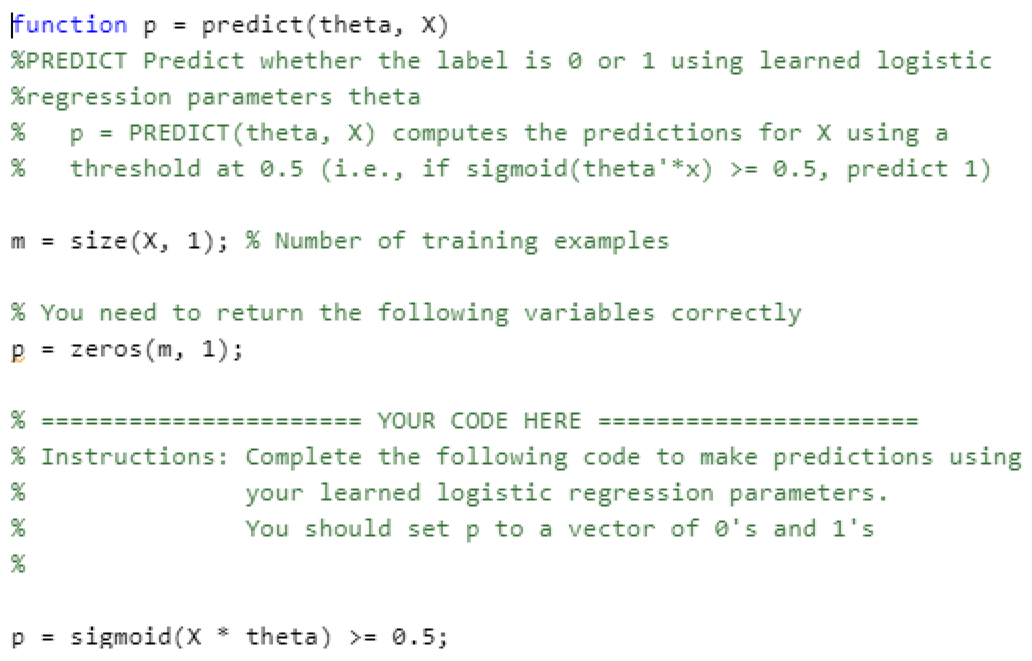
```
function plotData(X, y)
%PLOTDATA Plots the data points X and y into a new figure
%   PLOTDATA(x,y) plots the data points with + for the positive examples
%   and o for the negative examples. X is assumed to be a Mx2 matrix.

% Create New Figure
figure; hold on

% ====================== YOUR CODE HERE ======================
% Instructions: Plot the positive and negative examples on a
%               2D plot, using the option 'k+' for the positive
%               examples and 'ko' for the negative examples.
%

pos = find(y == 1);
neg = find(y == 0);

plot(X(pos, 1)), X(pos,2), 'k+','LineWidth', 2, 'MarkerSize', 7);
plot(X(neg, 1)), X(neg,2), 'ko', 'MarkerFaceColor', 'y', 'MarkerSize', 7);



% =========================================================================
```



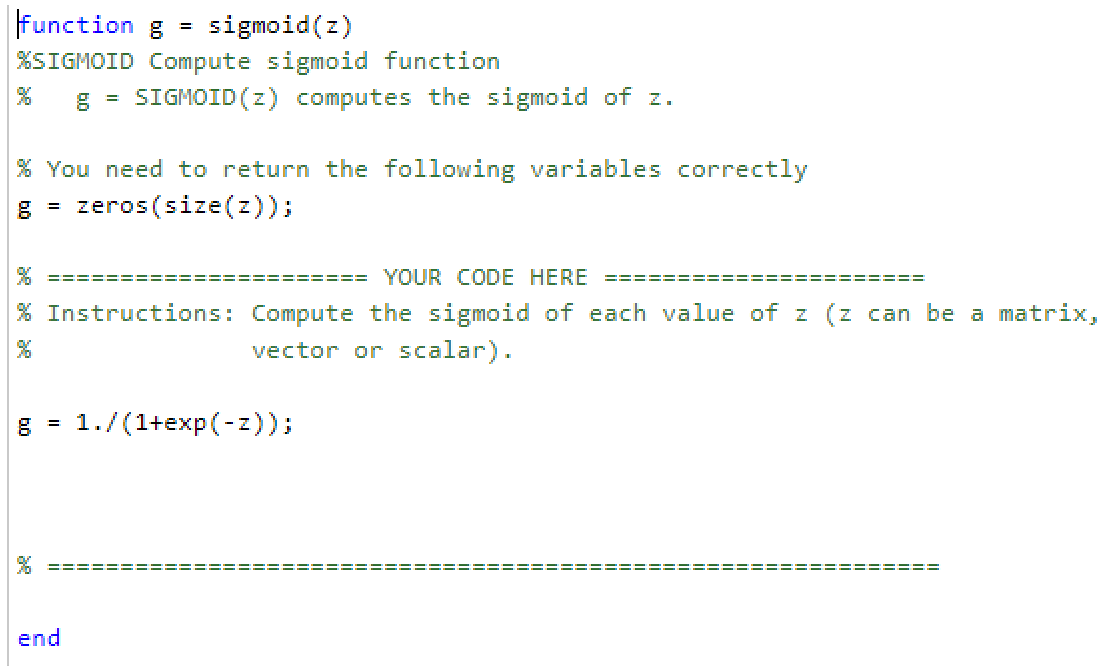
```
function [J, grad] = costFunction(theta, X, y)
%COSTFUNCTION Compute cost and gradient for logistic regression
%   J = COSTFUNCTION(theta, X, y) computes the cost of using theta as the
%   parameter for logistic regression and the gradient of the cost
%   w.r.t. to the parameters.

% Initialize some useful values
m = length(y); % number of training examples

% You need to return the following variables correctly
J = 0;
grad = zeros(size(theta));

% ====================== YOUR CODE HERE ======================
% Instructions: Compute the cost of a particular choice of theta.
%               You should set J to the cost.
%               Compute the partial derivatives and set grad to the partial
%               derivatives of the cost w.r.t. each parameter in theta
%
% Note: grad should have the same dimensions as theta
%
h = sigmoid(X * theta);
J = (1 / m) * sum(-y .* log(h) - (1 - y) .* log(1 -h));
```

```
grad = (1 / m) * sum((h - y) .* X);

% is equivalent to
h = sigmoid(X * theta);
J = (1 / m) * (-y' * log(h) - (1 - y)' * log(1 - h));
grad = (1 / m) * X' * (h - y);
```

```
function [J, grad] = costFunctionReg(theta, X, y, lambda)
%COSTFUNCTIONREG Compute cost and gradient for logistic regression with regularization
%   J = COSTFUNCTIONREG(theta, X, y, lambda) computes the cost of using
%   theta as the parameter for regularized logistic regression and the
%   gradient of the cost w.r.t. to the parameters.

% Initialize some useful values
m = length(y); % number of training examples

% You need to return the following variables correctly
J = 0;
grad = zeros(size(theta));

% ====================== YOUR CODE HERE ======================
% Instructions: Compute the cost of a particular choice of theta.
%               You should set J to the cost.
%               Compute the partial derivatives and set grad to the partial
%               derivatives of the cost w.r.t. each parameter in theta
% solution 1

h = sigmoid(X * theta);
left_part = -y' * log(h);
right_part = (1 - y') * log(1 - h);
theta_zero = theta;
theta_zero(1) = 0;
lambda_cost_part = (lambda / (2 * m)) * sum(theta_zero .^ 2);
lambda_gradient_part = (lambda / m) * theta_zero;
J = (1 / m) * sum(left_part - right_part) + lambda_cost_part;
grad = (1 / m) * (X' * (h - y)) + lambda_gradient_part;




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

end
```



```
function p = predict(theta, X)
%PREDICT Predict whether the label is 0 or 1 using learned logistic
%regression parameters theta
%   p = PREDICT(theta, X) computes the predictions for X using a
%   threshold at 0.5 (i.e., if sigmoid(theta'*x) >= 0.5, predict 1)

m = size(X, 1); % Number of training examples

% You need to return the following variables correctly
p = zeros(m, 1);

% ====================== YOUR CODE HERE ======================
% Instructions: Complete the following code to make predictions using
%               your learned logistic regression parameters.
%               You should set p to a vector of 0's and 1's
%

p = sigmoid(X * theta) >= 0.5;
```



```
function g = sigmoid(z)
%SIGMOID Compute sigmoid function
%   g = SIGMOID(z) computes the sigmoid of z.

% You need to return the following variables correctly
g = zeros(size(z));

% ====================== YOUR CODE HERE ======================
% Instructions: Compute the sigmoid of each value of z (z can be a matrix,
%               vector or scalar).

g = 1./(1+exp(-z));



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

end
```