```
function A = warmUpExercise()
%WARMUPEXERCISE Example function in octave
%   A = WARMUPEXERCISE() is an example function that returns the 5x5 identity matrix

A = [];
% ============= YOUR CODE HERE ==============
% Instructions: Return the 5x5 identity matrix
%               In octave, we return values by defining which variables
%               represent the return values (at the top of the file)
%               and then set them accordingly.


A = eye(5);




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


end
```

```
function plotData(x, y)
%PLOTDATA Plots the data points x and y into a new figure
%   PLOTDATA(x,y) plots the data points and gives the figure axes labels of
%   population and profit.

figure; % open a new figure window
plot(x,y,'rx', 'MarkerSize',10);
ylabel('Profit in $10,000s');
xlabel('Population of City in 10,000s');

% ====================== YOUR CODE HERE ======================
% Instructions: Plot the training data into a figure using the
%               "figure" and "plot" commands. Set the axes labels using
%               the "xlabel" and "ylabel" commands. Assume the
%               population and revenue data have been passed in
%               as the x and y arguments of this function.
%
% Hint: You can use the 'rx' option with plot to have the markers
%       appear as red crosses. Furthermore, you can make the
%       markers larger by using plot(..., 'rx', 'MarkerSize', 10);
```

```
function J = computeCost(X, y, theta)
%COMPUTECOST Compute cost for linear regression
%   J = COMPUTECOST(X, y, theta) computes the cost of using theta as the
%   parameter for linear regression to fit the data points in X and y

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

% You need to return the following variables correctly
J = sum(power(X*theta-y,2)/(2*m));

% ====================== YOUR CODE HERE ======================
% Instructions: Compute the cost of a particular choice of theta
%               You should set J to the cost.




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

end
```

```
function [theta, J_history] = gradientDescent(X, y, theta, alpha, num_iters)
%GRADIENTDESCENT Performs gradient descent to learn theta
%   theta = GRADIENTDESCENT(X, y, theta, alpha, num_iters) updates theta by
%   taking num_iters gradient steps with learning rate alpha

% Initialize some useful values
m = length(y); % number of training examples
J_history = zeros(num_iters, 1);

for iter = 1:num_iters
    theta1=theta(1,1)-alpha/m*sum(X*theta-y);
    theta2=theta(2,1)-alpha/m*sum((X*theta-y) .* X(:, 2));
    theta(1,1)=theta1;
    theta(2,1)=theta2;

    % ====================== YOUR CODE HERE ======================
    % Instructions: Perform a single gradient step on the parameter vector
    %               theta.
    %
    % Hint: While debugging, it can be useful to print out the values
    %       of the cost function (computeCost) and gradient here.
    %
```

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

    % Save the cost J in every iteration
    J_history(iter) = computeCost(X, y, theta);

end

end
```