```
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
%X = [ones(m,1),x];
%num_iters = 1500;
%alpha = 0.01;
J_history = zeros(num_iters, 1);

for iter = 1:num_iters

    % ====================== 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.
    %


temp0 = theta(1,1) - alpha/m*sum((X*theta-y));
temp1 = theta(2,1) - alpha/m*sum((X*theta-y).*X(:,2));
theta(1,1) = temp0;
theta(2,1) = temp1;

%temp = theta - alpha/m*X'*(X*theta-y)
%theta=temp;

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

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

end

fprintf('Theta computed from gradient descent:\n%f,\n
%f',theta(1),theta(2))
%plot(x,y,'rx','MarkerSize',5);
%ylabel('Profit in $10,000s');
%xlabel('Population of City in 10,000s');
%hold on; % keep previous plot visible
%plot(X(:,2), X*theta, '-')
%legend('Training data', 'Linear regression')
%hold off % don't overlay any more plots on this figure
% Predict values for population sizes of 35,000 and 70,000
```

```
predict1 = [1, 3.5] *theta;
fprintf('For population = 35,000, we predict a profit of %f\n',
 predict1*10000);
predict2 = [1, 7] * theta;
fprintf('For population = 70,000, we predict a profit of %f\n',
 predict2*10000);

end
```

```
Not enough input arguments.

Error in gradientDescent (line 8)
m = length(y); % number of training examples
```

Published with MATLAB® R2020b