function J = computeCostMulti(X, y, theta)

*%COMPUTECOSTMULTI Compute cost for linear regression with multiple variables*

*% J = COMPUTECOSTMULTI(X, y, theta) computes the cost of using theta as the*

*% parameter for linear regression to fit the data points in X and y*

m = length(y); % number of training examples

J = 0;

h=X\*theta;

err=(h-y).^2;

err=sum(err);

J=err/(2\*m);

End

-----------------------------------------------------------------------------------------------------------------------------

function [theta, J\_history] = gradientDescentMulti(X, y, theta, alpha, num\_iters)

%GRADIENTDESCENTMULTI Performs gradient descent to learn theta

% theta = GRADIENTDESCENTMULTI(x, y, theta, alpha, num\_iters) updates theta by

% taking num\_iters gradient steps with learning rate alpha

m = length(y); % number of training examples

J\_history = zeros(num\_iters, 1);

for iter = 1:num\_iters

h=X\*theta;

error=h-y;

T=transpose(X);

change=T\*error;

%changed=sum(change);

change=change\*((1/m)\*alpha);

theta=theta-change;