```
function [J grad] = nnCostFunction(nn_params, ...
                                   input_layer_size, ...
                                   hidden_layer_size, ...
                                   num_labels, ...
                                   X, y, lambda)



Theta1 = reshape(nn_params(1:hidden_layer_size * (input_layer_size + 1)), ...
                 hidden_layer_size, (input_layer_size + 1));

Theta2 = reshape(nn_params((1 + (hidden_layer_size * (input_layer_size + 1))):end), ...
                 num_labels, (hidden_layer_size + 1));

% Setup some useful variables
m = size(X, 1);

% You need to return the following variables correctly
J = 0;
Theta1_grad = zeros(size(Theta1));
Theta2_grad = zeros(size(Theta2));




%Feed-forward

regu = 0;
for i=1:size(Theta1, 1)
   for j=2:size(Theta1, 2)
      regu = regu + Theta1(i,j)*Theta1(i,j);
   end
end

for i=1:size(Theta2, 1)
   for j=2:size(Theta2, 2)
      regu = regu + Theta2(i,j)*Theta2(i,j);
   end
end

regu = regu*lambda/(2*m);

a1 = X;
a1 = [ones(m, 1) a1];
z2 = a1*Theta1';
a2 = sigmoid(z2);
a2 = [ones(m, 1) a2];
z3 = a2*Theta2';
a3 = sigmoid(z3);

y1 = zeros(m, num_labels);
for i=1:m
```

```
    for j=1:num_labels
      if j == y(i)
        y1(i, j) = 1;

      end
    end
end



for i=1:m
    J = J - log(a3(i, :)) * (y1(i, :)') - log(1-a3(i, :)) * (1- y1(i, :)');
end
J=J/m;

J=J+regu;




%Back propagation starts here
%gradient calculation
dl3 = a3 - y1;
dl2 = (dl3*Theta2).*(a2.*(1-a2));

ddl2 = dl2(:, 2:size(dl2,2));
Theta2_grad = Theta2_grad + dl3'*a2;
Theta1_grad = Theta1_grad + ddl2'*a1;
Theta2_grad = Theta2_grad/m;
Theta1_grad = Theta1_grad/m;

for i=1:size(Theta2_grad,1)
    for j=2:size(Theta2_grad,2)
      Theta2_grad(i, j) = Theta2_grad(i, j) + lambda*Theta2(i,j)/m;
    end
end

for i=1:size(Theta1_grad,1)
    for j=2:size(Theta1_grad,2)
      Theta1_grad(i, j) = Theta1_grad(i, j) + lambda*Theta1(i,j)/m;;
    end
end




% Unroll gradients
grad = [Theta1_grad(:) ; Theta2_grad(:)];


end
```