Your job is to implement the functions `train_perceptron`, `test_perceptron`, and `generate_moons` in Matlab; use them to carry out the computer experiment described by Haykin in chapter 1 of his text; and write a brief report (no more than two pages) describing what you did. To do this, you will need to generate a Haykin *moon* data training set, train a single layer perceptron using it, then generate a test set and evaluate it using the SLP weights.

The functions are to have the input and output arguments listed below.

```
function w = train_perceptron(w, X, d, eta, epochs, ...
     learning, display_x, display_y)
```

where

- Input `w` is a column vector containing m+1 initial weights,
- output `w` is a column vector containing m+1 trained weights,
- `X` is an mXN matrix containing N training samples each of which is a column of m values,
- `d` is a column vector containing N desired output values selected from {-1,1},
- `eta` is a learning rate value,
- `epochs` is the maximum number of epochs to iterate,
- `learning` is either `'sequential'` for sequential learning or `'batch'` for batch learning, and
- `display_x, display_y` (optional arguments, i.e., either neither are provided or both are provided) if provided, specify the range of x and y values over which to display the class confidence values (v values).

Given that the training set is a linearly separable collection of training vectors, the function should use the perceptron learning rule to identify weights that will allow a perceptron to correctly classify all elements of the training set.

If you want, you can make your learning rate change as a function of time.

The function must display the MSE on the training set at the end of each training epoch. The MSE should be calculated as follows:

$$MSE = \frac{1}{N} \sum_{x \in X} (\vec{w}^T \vec{x}\, d(x))^2 \quad .$$

Compute this over the set X of misclassified training vectors. (Note: the divisor N is the total number of training vectors.)

```
function y = test_perceptron(x,w)
```

where

- `x` is a column vector containing m values
- `w` is a column vector of m+1 weights, and
- `y` is the result of evaluating the single-layer perceptron with weights `w` and input `x`. (Note: `y` should have value 1 or -1.)

```
function [ X,d] = generate_moons(dist, r, w, N)
```

where

- Input `dist` is the *y*-axis distance (labeled *d* in the picture below) between the moon regions,
- `r` is the radius of the moons,
- `w` is the width of each moon,
- `N` is the number of samples to be generated,
- output `X` is a 2X*N* matrix containing samples from the moons, generated by uniformly sampling an angle (in range 0, 2π) and radius (in range r ±w/2),
- output `d` is an *N* element column vector containing value 1 if the point is in the top moon and value -1 if the point is in the bottom moon.