This practical requires you to use the MATLAB programming language to implement a set of specific features described below.

# Test-Driven Development

- Start by download and extracting the zip archive provided. The archive contains a set of MATLAB files, including some that you need to edit.

  The archive also contains a unit test file vectorAndMatricesTest.m that enables automated marking of your MATLAB code.

- To begin programming, start MATLAB and change the *MATLAB current folder* to the folder you just extracted from the zip archive.

- Enter the command

  ```
  runtests('vectorAndMatricesTest')
  ```

  in the MATLAB Command Window.

  This should produce the output shown in the in Figure 1, indicating that all the given tests have failed. This is because you havent yet implement the tasks.

  The error messages also include explicit feedback on why your tests failed and can be a very helpful development tool.

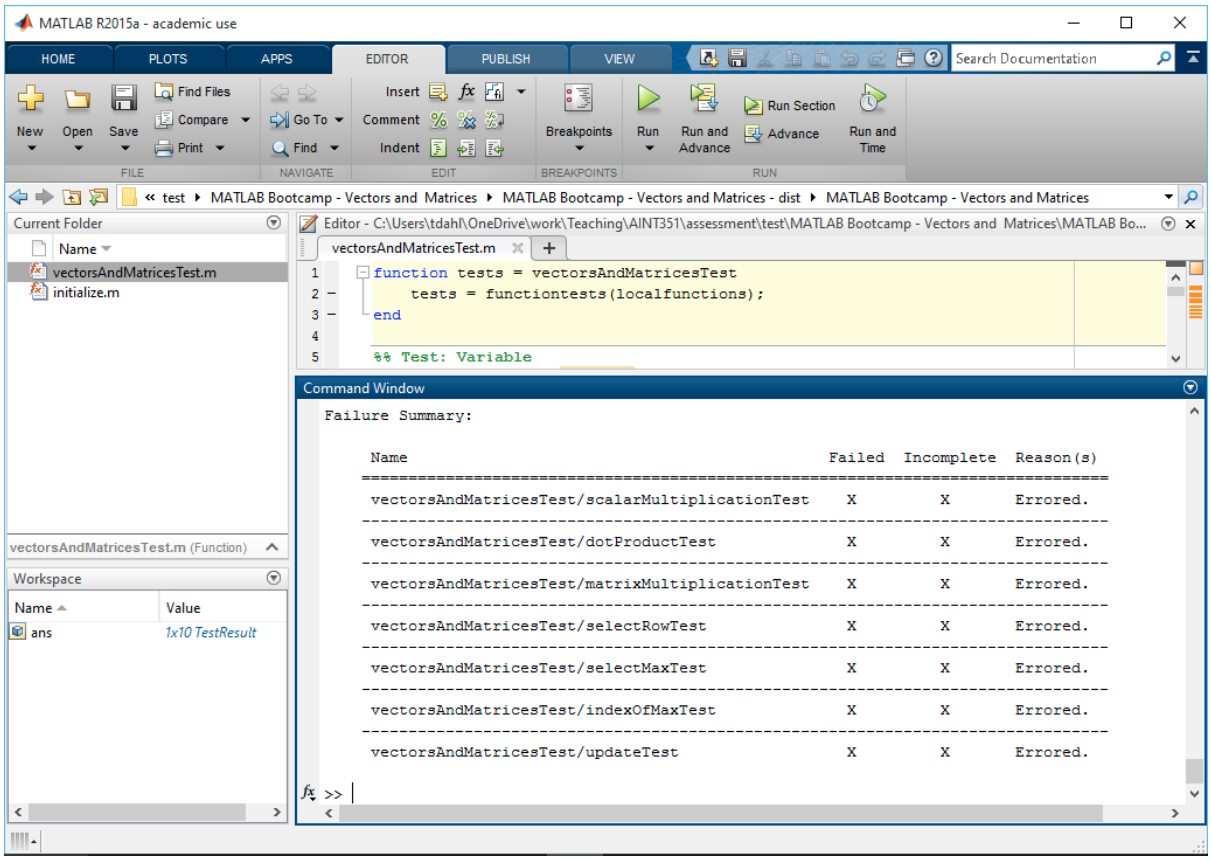


**Figure 1: The output from running the provided tests on the given files. Five failed tests are indicated.**

- You can run the tests at any time, and as you add code, the list of failed tests will get shorter until your code passes all the tests.

- The output when your code passes all tests is shown in Figure 2.

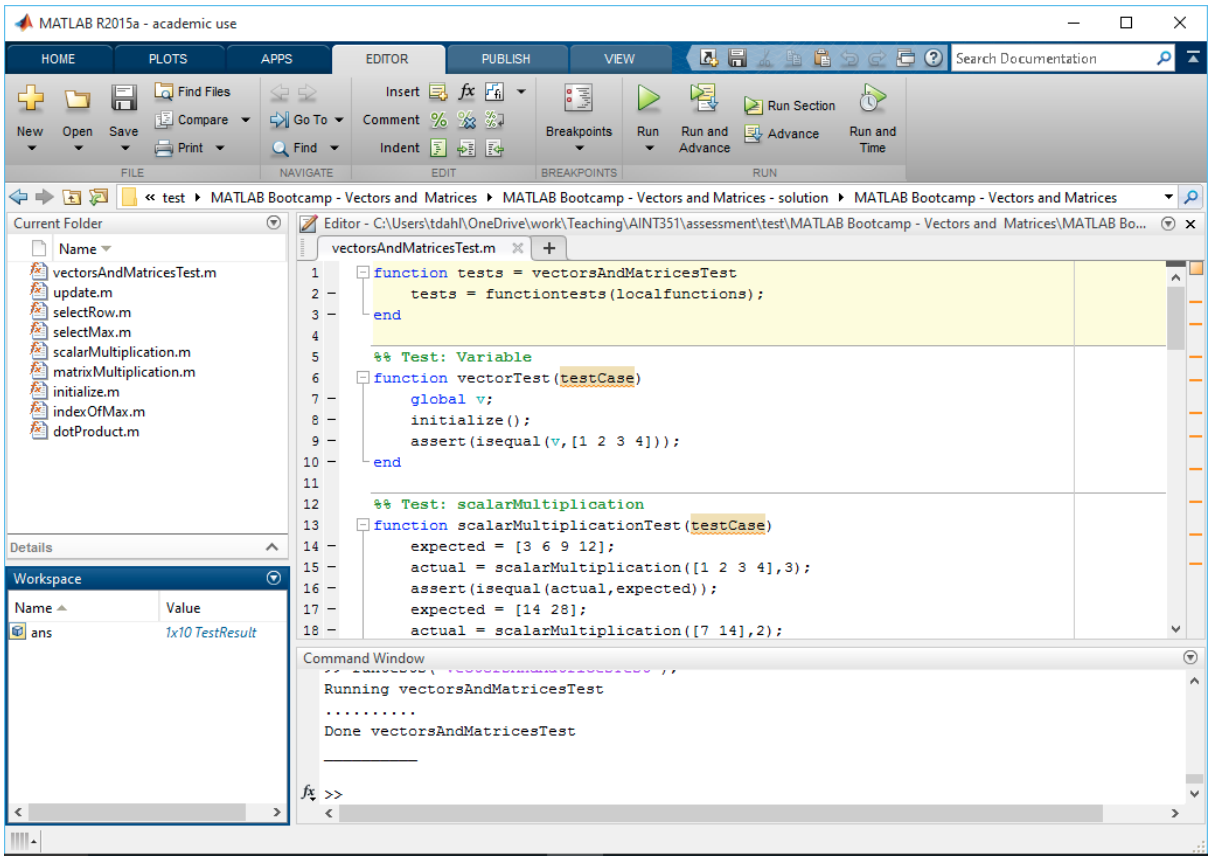


**Figure 2: The test output when your code satisfies all tests**

# Required Features

1. **Vector**: Add code to the initialize.m file so that it declares a global variable named v and initialises it to the row vector

$$[1 \quad 2 \quad 3 \quad 4]$$

2. **Scalar multiplication:** Add a file containing a function called scalarMultiplication(v, s).
   The function should take two arguments, a vector, v, and a scalar value, s.
   The function should return the result of the scalar multiplication of the scalar and the vector.

3. **Column vector**: Add code to the initialize.m file so that it declares a global variable named u and initialises it to the column vector

$$\begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$$

4. **Dot product**: Add a file containing a function called dotProduct(v, w). The function should take two arguments, both vectors, and return the dot product (scalar product or inner product) of the two vectors.

5. **Matrix**: Add code to the initialize.m file so that it declares a global variable named M and initialises it to:

$$\begin{bmatrix} 2 & 4 & 6 & 8 \\ 10 & 12 & 14 & 16 \\ 18 & 20 & 22 & 24 \end{bmatrix}$$

6. **Matrix multiplication:** Add a file containing a function called matrixMultiplication(M, N).
   The function should take two arguments, both matrices, and return the product of the two matrices.

7. **Row selection:** Add a file containing a function called selectRow(M, r). The function should take two arguments, a matrix and a scalar value, and return the row that has the given scalar as its index (starting from 1).

8. **Maximum value selection:** Add a file containing a function called selectMax(M).
   The function should take a row vector as an argument and return the maximum value of the vector.

9. **Maximum value index:** Add a file containing a function called indexOfMax(M).

The function should take a row vector as an argument and return the index of the maximum value.

10. **Maximum value update:** So finally, given a matrix and a row index, we want to update the maximum value of that row in this case by multiplying it by 1.2.
    Add a file containing a function called updateMax(M,s).
    The function should take two arguments, a matrix and a row index.
    The function should return a copy of M where the maximum value in the given row has been multiplied by 1.2.

    For example, given the matrix M1 below:

$$M1 = \begin{bmatrix} 0.3 & 2.2 \\ 2.0 & 1.5 \end{bmatrix}$$

    Let's say we want to update the maximum value of row 2. If the matrix M2 i be returned by the updateMax function, we should get he result

$$M2 = \begin{bmatrix} 0.3 & 2.2 \\ 2.4 & 1.5 \end{bmatrix}$$