AINT351 Machine Learning 2018
Lab practical 1A: MATLAB Variables and Control

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

The activities are based loosely on traditional text adventure games where a user interacts with a virtual world by entering commands and reading messages describing the world.

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 variablesAndControlTest.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.

- As indicated by the Matlab help, the runtests function provides a simple way to run a collection of tests.

  Calling the runtests function as follows:

  ```
  RESULT = runtests(TESTS)
  ```

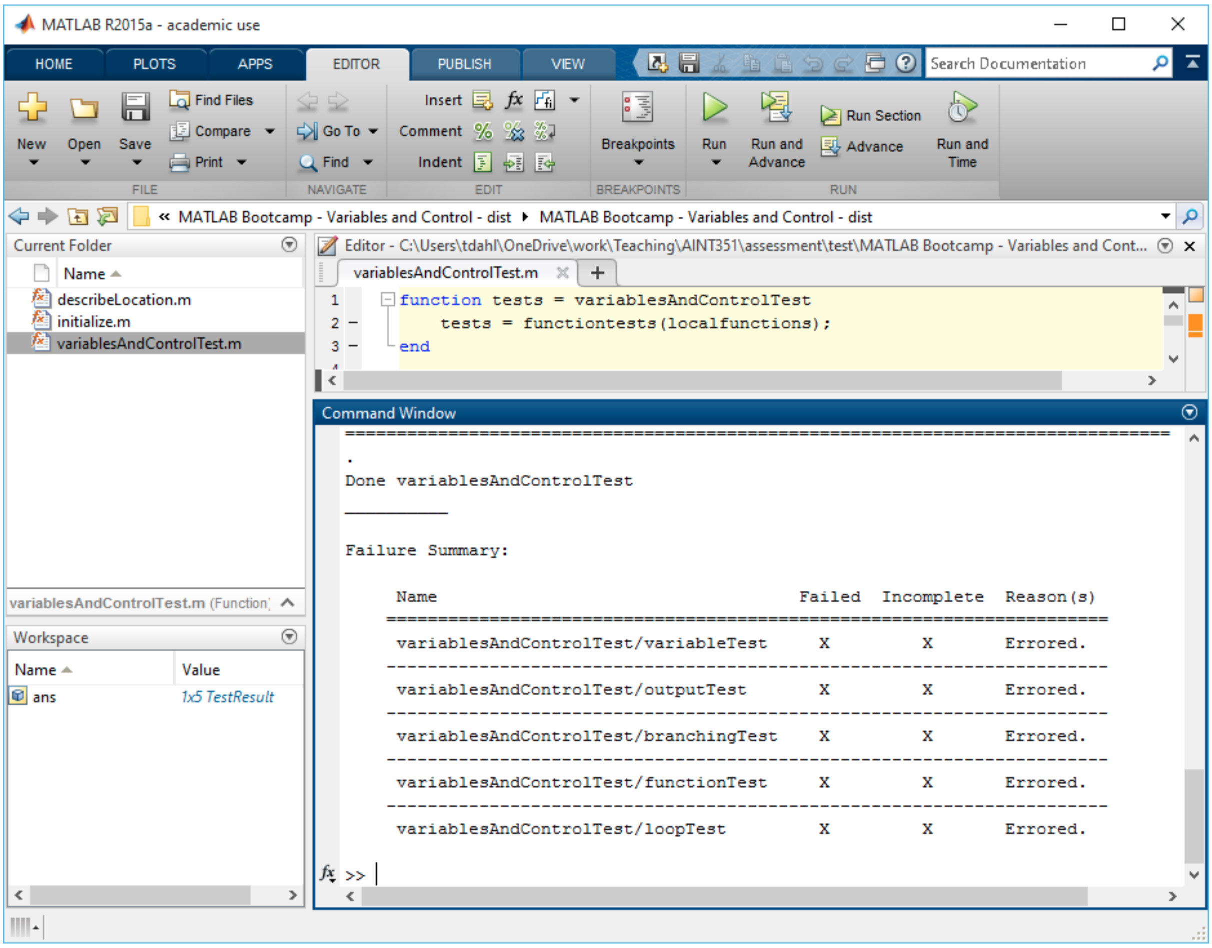
  creates a test suite specified by the function TESTS, runs them, and returns the RESULT.

- We will now run the unit tests for the laboratory task. Enter the command

  ```
  runtests('vectorsAndMatricesTest')
  ```

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.


MATLAB R2015a - academic use
HOME
PLOTS
APPS
EDITOR
PUBLISH
VIEW
Search Documentation
New
Open
Save
Find Files
Compare
Print
FILE
Go To
Find
NAVIGATE
Insert
Comment
Indent
EDIT
Breakpoints
BREAKPOINTS
Run
Run and Advance
Run Section
Advance
Run and Time
RUN
« MATLAB Bootcamp - Variables and Control - dist ▸ MATLAB Bootcamp - Variables and Control - dist
Current Folder
Name
describeLocation.m
initialize.m
variablesAndControlTest.m
variablesAndControlTest.m (Function)
Workspace
Name
Value
ans
1x5 TestResult
Editor - C:\Users\tdahl\OneDrive\work\Teaching\AINT351\assessment\test\MATLAB Bootcamp - Variables and Cont...
variablesAndControlTest.m
Command Window

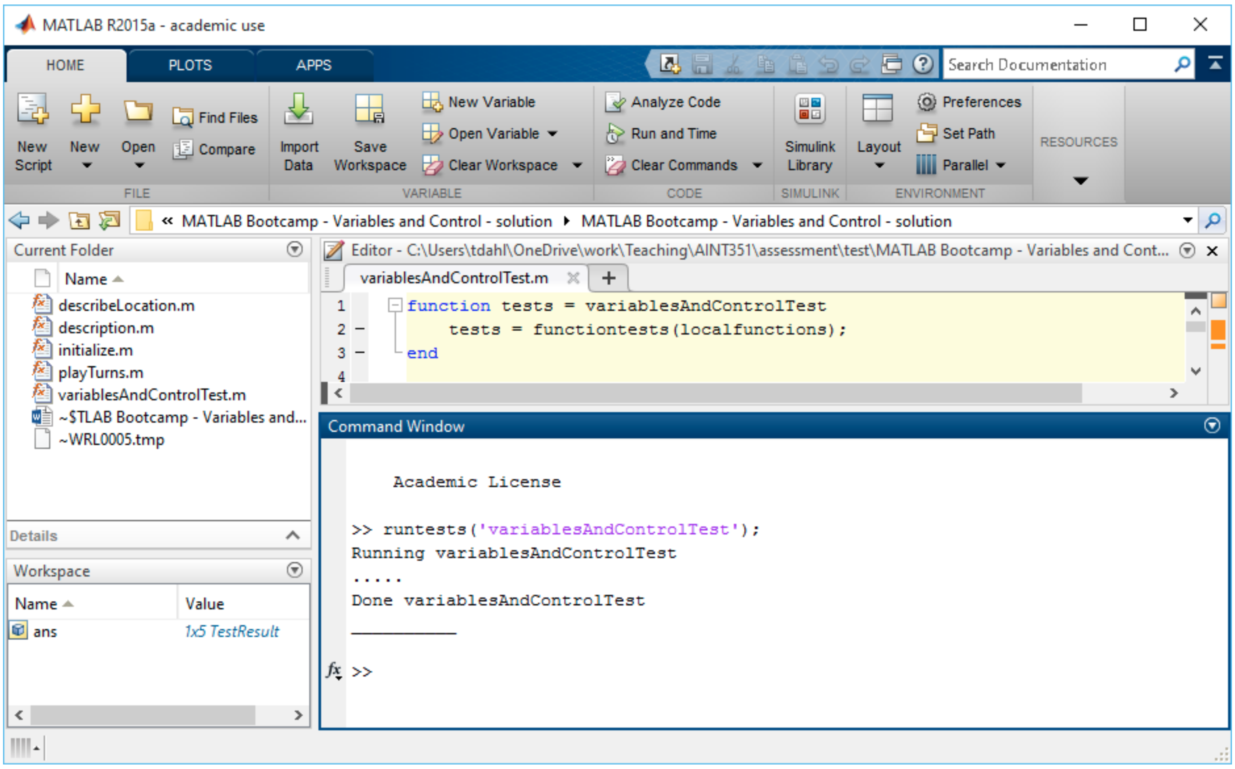
```
function tests = variablesAndControlTest
    tests = functiontests(localfunctions);
end
```

```
================================================================================
.
Done variablesAndControlTest
__________

Failure Summary:

     Name                                     Failed  Incomplete  Reason(s)
    ====================================================================
     variablesAndControlTest/variableTest       X         X       Errored.
    --------------------------------------------------------------------
     variablesAndControlTest/outputTest         X         X       Errored.
    --------------------------------------------------------------------
     variablesAndControlTest/branchingTest      X         X       Errored.
    --------------------------------------------------------------------
     variablesAndControlTest/functionTest       X         X       Errored.
    --------------------------------------------------------------------
     variablesAndControlTest/loopTest           X         X       Errored.

>>
```

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.



```
function tests = variablesAndControlTest
    tests = functiontests(localfunctions);
end
```

```
   Academic License

>> runtests('variablesAndControlTest');
Running variablesAndControlTest
.....
Done variablesAndControlTest
__________

>>
```

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

Required Features

1. **Variables**: Add code to the initialize.m file so that it declares a global variable named playerLocation and initialises it to the value 2.

2. **Output**: Add code to the describeLocation.m file so that it prints out the text ‘You are in a sunny field’.

3. **Branching**: Extend the code in the describeLocation.m file so that the text printed depends on the value of the playerLocation variable. The text should always start with ‘You are’. This should be followed by a location description as illustrated in the map of the Dungeons of Doom world given in Figure 3. If the location variable is anything other than 0, 1, 2 or 3, the text output should be ‘You are lost’. Note that your code will not pass this test until all the messages are printed out correctly.

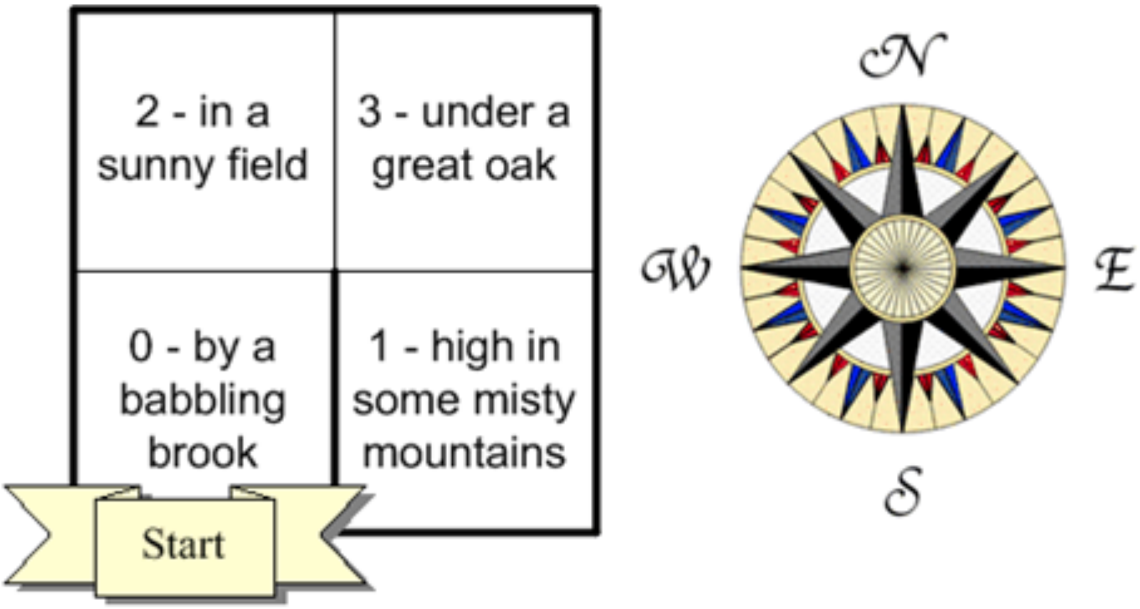


Figure 3: Map of the Dungeons of Doom world

4. **Functions:** Copy the code from the describeLocation.m file into a new file called description.m. Add a function decription() to the new file that takes a single argument and returns a single value. Copy the code from the describeLocation.m file into the description function and modify it so that the function returns the description string corresponding to the value given in the function parameter.

5. **Loops:** Finally, add another file and function playTurns that takes a single argument. This function should print out the description of the current location as many times as indicated by the function parameter, i.e., if the parameter's value is 100 and the value of the playerLocation variable is 3, the text 'You are under a great oak' should be printed out 100 times.