Cog Sci 242 **Lab 1: Reverse Engineering** *Fall 2018*

The theme of this first lab is "reverse engineering."

Working with a partner, you will be presented with a variety of "black boxes," systems that accomplish something but whose workings are initially hidden from you.

In each case, you will be challenged to figure out how it is working. Then you will be able to "open" the black box and see if you are correct.

Along the way, you will be introduced to the programming language MATLAB.
We will be using this language quite a bit in the labs for this course.

<u>About MATLAB</u>

The name "MATLAB" is short for "Matrix Laboratory." A matrix is a rectangular array – a table. Matrices turn out to be a very useful structure in solving many problems in scientific computation. MATLAB was originally developed by applied mathematicians who specialized in matrix calculations. It has now evolved into a general-purpose programming language that is widely used by scientists and engineers.

<u>Running MATLAB in the Fowler Computer Labs</u>

1. Log into your Oxy account.
   (If you are working with a partner, one of you should do this.)

2. If you have not already done so, create a MATLAB folder on your G: drive into which you will place the programs that you write. This folder should be directly off of your root directory, and hence have the path

   G:\MATLAB

2. Click on the "Start" icon at the lower left corner of your screen, and select:

   All Programs → Cognitive Science → MATLAB Cog Sci 242

3. MATLAB will load. When it has done so, you will see a variety of windows and menu bars. The **large window in the center** called the **Command Window.**

   At the left margin of the Command Window you will see the following:

   >>

   This is called a "prompt" – it means that MATLAB is ready for you to give it instructions by typing them to the right of the prompt.

Telling MATLAB What You Want It to Do

To communicate with MATLAB via the Command Window you type *commands*.

A command may be something chosen from MATLAB's existing library of commands or it may be something you have created yourself.

In either case, the command has to be used precisely – if it isn't, you will see an error message and have to try again. (If you are new to programming, don't let this distress you. Programming is definitely a trial-and-error thing. *Everyone* typically makes many errors before ultimately succeeding.)

Getting Started

In the Command Window (at the center of the screen), type the *path* command, then press **[Enter]**. The **[Enter]** key must always be pressed to execute a command line.

```
>>path
```

You will see many lines scroll by, each of which is listing the **pathname** of a directory on your computer. These are directories where MATLAB will look for the instructions it should follow when it is given a command.

The last two lines are:

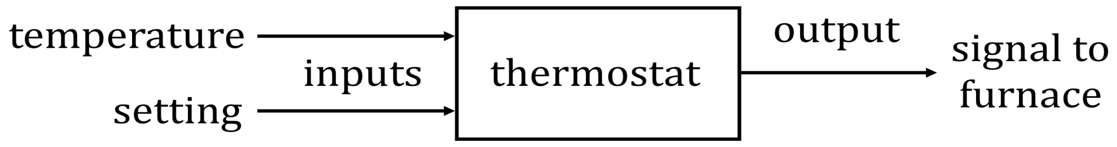
```
S:\Cognitive Science\MATLAB\Cog Sci 242

G:\MATLAB
```

This tells you that you will be able to read any programs that are found in our course directory on the S: drive, and also that your own MATLAB folder on your G: drive is your current working directory. (This is where programs you write will be stored by default.)

Black Boxes

You may recall this “computational level” diagram of a thermostat from the *Class 1* powerpoint slides:



This diagram tells you *what* a thermostat does in very broad terms (it takes “temperature” and “setting” as inputs, and outputs some kind of a signal to a furnace) but it doesn’t tell you *how* the thermostat works.

In this diagram the thermostat is a “black box” – a box we can’t see inside.

You will now work with a series of MATLAB programs I’ve written that will initially be black boxes to you.

In each case, by studying the way the program behaves (i.e. what sort of output results from giving it certain inputs) you will develop an idea about what the program does and can speculate on how it does it.

In each case, you will then be able to “open up the box” and see what’s actually going on. In addition to checking to see if you were correct, you will start learning some of the MATLAB language.

Running and Analyzing Scripts

A **script** in MATLAB is a file that contains a sequence of commands that you could have entered in the command window. When you run the script, the commands are executed in the order they occur in the file. NOT ALL programs in MATLAB are scripts, but they're the easiest type of MATLAB program to start with.

To run a script in MATLAB, all you do is enter its name at the prompt.

You will now run two scripts that are located in our course directory.
These will serve as "black boxes" that you will attempt to reverse-engineer.

After running each script, you will first write an account of *what* it is doing – i.e. a description of it at the *computational level.*

Then you can try to figure out *how* it is doing what it does. That is, you will attempt an *algorithmic* description of the script.

(This does not involve MATLAB per se – though the more you know about MATLAB the more likely you are to propose an algorithm close to the one being used.

In the future, when you know more MATLAB, you could take the last step and propose a MATLAB *implementation* of your algorithm.)

Finally, you will look at the actual script file. Based on this, you will be able to

- revise your algorithmic description so that it matches what MATLAB is doing,
- and see how that algorithm is *implemented* in the MATLAB language.

Subsequent pages in this lab handout will help you understand in detail the different lines in these scripts.

CHECK POINT

Any Questions?

This is a good point to stop and make sure you understand what you are about to do. Ask me questions if you don't!

Reverse-Engineering BlackBox1

Run the script *BlackBox1* by entering its name after the prompt:

```
>>BlackBox1
```

Record your analyses of this script below:

Computational Analysis of BlackBox1

Algorithmic Analysis of BlackBox1

To read the actual script file:

```
>>type BlackBox1
```

Compare the *comments* in this file to your analyses. The top comments essentially give a "computational level" description, while the remaining comments describe the algorithm along with some aspects of its implementation.

MATLAB Features

Comments

A line that begins with a % is a comment line. This is a line that people can read, but that the computer ignores.

Assignment

In MATLAB a single "=" indicates an *assignment*. (The value of the expression to the right of the "=" is assigned to the variable to the left of the "=".)

For example, if you saw the following lines in MATLAB, what value would be stored in the variable z? (Try this in MATLA B.)

```
>> x = 3;
>> y = 2*x;
>> z = y/3;
>> z
```

Answer: 2

Using a Semicolon to Suppress Output to the Screen

If you read through the *BlackBox1* script, you will notice that some lines end with a semicolon (;) while others do not.

When a semicolon appears at the end of a line, the result of executing the line is NOT displayed on the screen. The calculation specified by the line is performed, but you don't see the result on the screen.

If a line does *not* end in a semicolon, then the result of executing the line *is* displayed.

Try entering each of these two lines at the prompt and compare what happens:

```
>> 2 + 4;
>> 2 + 4
```

Variables

A *variable* is a named memory location. Most of what a program does is specify how information will be stored in variables, read from variables, modified in some way, and then stored variables again. There are different types of variables in MATLAB according to the type of information stored in them.

Numerical Variables

Many computations result in a number. A variable storing a number is called a **numerical variable**. MATLAB has different classes of numerical variables based on how much memory is required to store the number, but you rarely have to be concerned about that.

- Which variables in *BlackBox1* are numerical?

Logical Variables

A **logical variable** has one of only two values, denoted in MATLAB by the reserved keywords

false **true**

or encoded by the numerals

0 1 .

- Which variables in *BlackBox1* are logical?

Note: Despite appearances, logical '0' is not the number 0, and logical '1' is not the number 1. Nonetheless, in some computations involving both numbers and logical values, these logical values will be treated like numbers.

- Can you find a line in this script where logical values are treated like numbers?

____

Answers: There is one numerical variable, *Select*. There is one logical variable, *TF*.
In the line *Select = 1 + TF* the logical values are treated like numbers.

String Variables

A sequence of adjacent characters (which may include whitespace characters), treated as a single object, is called a *string*. You can *designate* a sequence of characters as a string by placing the sequence between single quotes. Several strings can be concatenated into one string by placing them in the desired order within square brackets, [ ].

A **string variable** is a variable to which a string has been assigned.

- Give an example of a string found in the script *BlackBox1*.

- Which variables in *BlackBox1* are string variables?

- Give an example of using [ ] in *BlackBox1* to concatenate strings.

Cell Arrays

An array a rectangular table or collection of tables with the same dimensions. Note that it is possible to have a table with just one row, so that it looks like a list.

A *cell array* can be thought of as an array of boxes called *cells*. Among other things, you can put strings in cells, which makes a cell array useful when you want to make a table of strings.

To designate a collection of strings as a cell array, place the strings or the variables denoting them between curly braces, { }.

- Which variables in *BlackBox1* are cell arrays?

To *reference* a cell in a cell array, place its coordinates in the table between curly brackets after the name of the array. In *BlackBox1*, that is done in this line:

```
disp( Response{ Select } )
```

____

Answers: One example of such a string is 'Am I right? (Y/N):' .
There are three string variables: *Name*, *L* and *YN.*
['The name ' Name 'has ' L ' letters in it.'] is an example of string concatenation.
There is one cell array, *Response*.

This script is making use of several of MATLAB **functions**. In general, a MATLAB function is "called" as:

[list of output variables] = FunctionName(list of input variable values)

The variables in each of the ordered input and output lists are separated by commas.

It is possible to have functions with no explicit output (they do things, but don't ultimately store any results in memory) as well as functions with no explicit input.

For example, in *BlackBox1*, the line

```
Name = input('Please tell me your first name: ','s')
```

is a "call" to the function **input**.

The first input argument for this function is the string

'Please tell me your first name:'

The second input argument for this function is also a string:

's'

is a code telling MATLAB to output as a string whatever the user now inputs.

The function **input** then stores the user input as a string (because of the 's') in the output variable indicated – in this case the string variable *Name*.

- How do we know that *Name* is a string variable?

How do I know all this? Since **input** is one of MATLAB's built-in functions, it is documented at the MATLAB website, http://www.mathworks.com/help/.

You can type the name of a function you want to learn about in the search box at the upper right-hand corner of this web page.

- Do this now, and compare the documentation for the function *input* with our discussion above.

Answer: We know that *Name* is a string variable because *input* outputs a string when its second argument is the code 's'.

- Use the *Mathworks* website and what you have learned from *BlackBox1* to complete the following tables:

MATLAB function	*Output Variables & Types*	*Input Variables & Types*
str = **input**(*prompt*, 's')	*str* is a string variable	*prompt* is a string
disp		
length		
num2str		
strcmp		

MATLAB function	*What the Function Does*
str = **Input**(*prompt*, 's')	prompts the user for input and stores that input as a string in the designated string variable (that's what the 's' is encoding)
disp	
length	
num2str	
strcmp	

CHECK POINT: Ask me over to check your work
when you have completed the tables above.

Global and Local Variables

The script you ran, *BlackBox1*, had a lingering effect that you may not have been aware of.

To see what this effect is, enter each of the following at the prompt.
(Be sure that you do NOT use a semicolon at the end of these lines.)

>> Name

>> L

>> YN

>> TF

>> Select

>>Response

All the variables created by *BlackBox1* are still in your workspace after this script has finished running! (You'll also see these variables and their current values in the "Workspace" window at the lower left corner of the MATLAB screen.)

This is an important property of scripts – variables created by scripts are *global*, meaning that they exist in the workspace itself, not just within the script. A script can also make use of any global variables that exist in the workspace when it is run.

If you are running a number of scripts, your workspace can get pretty cluttered, and it can be difficult to keep track of all the global variables. MATLAB has a solution to this problem. Enter the following at the prompt.

>>clear Name L YN TF Select Response

Then enter (again with no semicolon):

>> Name, L, YN, TF, Select, Response

- What has happened to the variables?

You can design a script to clean up after itself in this way. Run the following script:

>>BlackBox2

then, once again,

>> Name, L, YN, TF, Select, Response

- Based on what you observe, what do you think the last line of *BlackBox2* is?

To check your guess,

>>type BlackBox2

<u>Your Turn</u>

Create a script called *MyFirstScript*. You can do this by first going to the menu bar at the top of the "Home" tab and selecting

Open → Open ...

then scrolling through the directories in the window at the left to find the file

S:\Cognitive Science\MATLAB\Cog Sci 242\BlackBox2.m

Note: The "S" drive may be called "AcadShared" in your window.

Then, from the menu bar at the top of the "Editor" tab in the Editor Window, select

Save → Save As ...

Then change the directory at the top of the "Select File for Save As" window to your own directory on the G: drive, and also change the *File name* to

MyFirstScript.m

before clicking on **[Save].**

You have now saved your first MATLAB file in your G:\MATLAB directory.

Now edit your *MyFirstScript* file so that it prompts the user to answer "Y," "N," or "Maybe" to the question "Am I right?"

In creating this script, you will need to figure out a response to "Maybe," and also how to have MATLAB produce that response.

You should also change the name of the script in the comments, add your name(s) as co-author(s), change the "Last Update" date, and modify the comments within the script to reflect what it now does.

CHECK POINT: Ask me over to show me your *MyFirstScript* script when you have it running. Feel free to ask me or other students for assistance if you are having trouble figuring it out or getting it to run properly.

Remember to edit the comments to correctly reflect what this script does and your co-authorship.

Quitting MATLAB

When you are done with a MATLAB session, you quit by simply entering the command *quit* at the prompt:

>>quit

Submitting Work

When you are ready,

upload your script file to Moodle using the submission folder provided.

Be sure that your script comments identify you and your partner.

Follow instructions in the submission folder for naming your file before uploading it.

Wrapping Up

This lab has introduced you to a number of programming concepts as well as some features specific to MATLAB.

Before you go, discuss the following questions with your partner and record your answers for future reference.

1. How do you start MATLAB for this course on the lab computers?

2. How do you open a MATLAB file (an “m file”) from within MATLAB in order to edit it?

3. What is a “string,” and how do you indicate in MATLAB that a particular sequence of characters is to be treated as a string?

4. What is a “logical variable”? What values can it have (in MATLAB)?

5. What does “case sensitive” mean? Is MATLAB case sensitive?

6. How would you code the following instruction in MATLAB?

“Assign the value 3 to the variable X.”

7. Which MATLAB functions would you use for the following purposes?

- comparing two strings to see whether or not they are the same
- prompting the user for input from the keyboard and assigning that input to a variable
- displaying a string on the screen
- converting a number to its corresponding string of numerals
- determining the length of a string

8. How do you prevent the result of a MATLAB command from being displayed?

9. How do you indicate that a line in a MATLAB program is a “comment” ?

Check your answers to these questions in the Lab 1 Notes file on Moodle.