Week 5 – Loops

Important: You should be able to do all questions before Exercise 4.1 without the help of the tutor, because these exercises are just copying examples from lectures 5-1 and 5-2.

1 Basic while Loop Questions

1.1 Printing 1 to 3 with while

To start this exercise, use to go to the folder where you saved your work from last week, open one of the files, click then Save As, and now save it in a new folder called lab5 and use the filename lab5_q1_1.m

Write a MATLAB program to print only the integers 1 to 3 (including the 1 and 3), using a while loop. Each of the numbers should appear on a separate line (on top of each other).

- Since this is your first time dealing with loops, you need to write comments describing each line of code, just like in lecture 5-1, slide 8. Do this now, so that you can just copy for the next few exercises and not have to type it all again.
- Make sure to indent correctly using the TAB key. Do not indent using the space key.

1.2 Printing between two other numbers

With your previous exercise open, click then Save As, and now use the filename lab5_q1_2.m This is what you should do each time you start a new exercise, so that you save your work in a new .m file and save time.

Modify your solution from exercise 1.1 so that instead of 1 to 3, you print the integers 5 to 21.

1.3 Skipping some numbers

Modify your solution from exercise 1.2 so that you print every 3rd integer, that is, you print 5, 8, 11, 14, 17, and 20.

1.4 Going backwards

Modify your solution from exercise 1.3 so that you print you print 50, 45, 40, 35, 30, 25, 20, 15, 10, and 5 in this order. Do not print the 0.

If your program does not stop, do not call the tutor. You should look at lecture 5-1 and figure out for yourself why this is occurring and how to fix it.
2 Basic for Loop Questions

2.1 Printing 1 to 3 with for
Print the integers 1 to 3, using a **for** loop.
- Use lecture 5-2 to help you
- Again, write comments describing each line of code
- Make sure to indent correctly using the TAB key

2.2 Printing other numbers
Modify your solution from exercise 2.1 so that you print 5, 8, 11, 14, 17, and 20, using a **for** loop. Since these numbers fit a pattern, you should not use the square brackets, you should use the colon instead.

3 Looping through Arrays Questions

3.1 Printing elements of an array
First make an array containing these 6 numbers in this same order:

\[
8, 4, 12, 9, 5, 10
\]

Then print each of the numbers in the array, using a **for** loop. Each of the numbers should appear on a separate line (on top of each other), and *not* beside each other.

3.2 Printing elements of an array with while
Do the same as in exercise 3.1, but use a **while** loop instead.

**Important**: When looping through arrays, you should *not* write code where you use a raw number when deciding how many times the loop should run:

```matlab
% Do NOT write code like this!
while i <= 6
```

Instead, you should use the **length** function, as shown in lecture 5-2, slide 53:

```matlab
% This is the correct way to use the number of elements % in an array
while i <= length(numbers)
```

This is correct (and using the raw value is bad) because it works no matter how many elements there are in the array. You should be getting Matlab to do the work and count how many elements there are in the array, you should not be counting and doing the work yourself.
3.3 Finding total
Modify exercise 3.1 not 3.2 (using the for loop, not while loop) and find the total of all the numbers in the array. Do not use the sum function (we have not learnt about this yet). Your program should only print the total to the screen, and should only print it once.

Important: You should have reached up to this point in 40 mins with virtually no help from the tutor. Do you see how the previous exercises are just copying the lecture notes?

If you needed significant help from the tutor on the previous exercises, or you were just copying the tutor, or you could not get to this point within 40 mins, then this is a very big warning sign that you must do things differently in future to pass this course. You cannot keep doing what you are doing. See lecture 2-1, slides 4-5

We provide you with every opportunity to pass this course, but it is up to you. It is your responsibility. We cannot force you to pass this course.

4 More Complex Looping Questions
4.1 Finding average
Modify exercise 3.3 and find the average of all the numbers in the array. Do not use the sum or average functions. Print only the average to the screen, and only print it once.

Important: As before, you should not write code where you divide by a raw number like this:

```
% Do NOT write code like this!
average = total / 6;
```

Instead, you will have to write code that figures out how many elements there are in an array. Of course, that was in the lectures. Go find it.

4.2 Finding average correctly
Modify only one line of code in exercise 4.1 so that instead of creating an array with 6 numbers you create an empty array, using code something similar to this:

```
% Create an empty array
numbers = [];
```

Now, run your program. You will see that the program gives a strange result:
**NaN** means “Not a Number”. Look at your code and think about what has happened to understand what this means so that you can understand how to fix it.

Now modify the rest of your program (but you must still keep the array empty) so that if the array is empty, your program does not give such strange results. Instead, you should print a nicer error message like “No average, since array is empty”. However, if the array is not empty, your code should print the average value correctly without printing this error message.

That is, whether you create an empty array or a non-empty array of numbers at the start of your program, the rest of your program still works correctly either way.

Do **not** use the **sum** or **average** functions. Your program should only print the average to the screen, and should only print it once.

### 4.3 Finding average of some numbers

Modify exercise 4.2 so that it only finds the average of those numbers greater than or equal to 10. Your program should only print this average to the screen, and should only print it once.

Your code should work correctly no matter what the array contains. If there were no numbers greater than or equal to 10, your program should print an appropriate message.

### 5 Harder Questions

**Remember**, questions in Lab Exam 2 will be similar in style and difficulty to the hardest questions in the labs.

#### 5.1 Finding maximum

First make an array containing the same numbers as in Exercise 3.1

Then write code using a loop so that it prints out the maximum value in the array.

Do **not** use the **max** function. Your program should only print the maximum value to the screen, and should only print it once.

Your program should work so that no matter what array is created at the start, you still print out the correct answer. If the array is empty, then your program can simply print “No maximum, since array is empty”.

This is a tricky Q, but using just what you have learnt, you should be able to do it. When your tutor demonstrates the solution, you will be kicking yourself at how simple it is, so try to do it now before they show you.
5.2 Checking array contents – the smarter way

Consider the code from last week’s lab (Lab 4), Exercise 3.1.

If there were 100 containers, then the code we wrote last week would obviously be very bad, because this part would very long:

```matlab
% Check if any container is boiling
if temps(1) >= 100 || temps(2) >= 100 || % and so on...
```

So obviously, the above way of doing it (which we did last week) is not flexible, and is therefore very bad. Now that we have learnt about loops, we should do it a smarter way.

Do the same exercise as last week, but this time, use a `for` loop to go through each temperature, thus avoiding the very long `if` statement like shown above. You should only use one loop.

Remember, last week’s exercise says to print `boiling` if there is at least one container boiling. That means you should only print `boiling` once, even if you find many containers boiling. Same idea for `freezing` and `all equal`.

Hints:

- Inside the loop:
  - Count how many containers are boiling
  - Count how many containers are freezing
  - Count how many containers have the same temperature as the first container
- Then after (outside) the loop:
  - Print appropriate messages depending on each of the above 3 counts

5.3 Inserting between elements into arrays – the smart way

Consider last week’s lab (Lab 4), Exercise 4.4.

If there were many elements to the right of where we want to insert a new element into the array, then there would be too much code repetition, which would lose marks, as shown in lecture 4-1, slide 25.

So obviously, the above way of doing it (which we did last week) is not flexible, and is therefore very bad. Now that we have learnt about loops, we should do it a smarter way.

Do the same exercise as last week, but this time, use a `for` loop and therefore avoiding code repetition. You should only use one loop.
5.4 Printing rectangles

Write a program, using only 2 `for` loops, and only typing the `$` character once in your program (because `$` is very expensive, you can only afford to type it once), to print the following rectangle on the screen:

```
$$$$$
$$$$$
$$$$$
```

To do this exercise, your code must have this line of code, so copy it as well as the comment (this is the only line of code that should use the `$` character):

```c
% Print just one $ character. The next character that is
% printed will appear on the same line, because we didn't
% use the \n (new line character) here.
fprintf('$');
```

This is using the `fprintf` function, which we will learn more about in week 8. You will also need this line of code (copy this code and the comment):

```c
% Print a new line character. This is like pressing enter
% on the keyboard. The next character that is printed
% will appear on the next line below
fprintf('\n');
```

Hints:

- A loop is used to repeatedly do something many times
- You need to print many characters on a line. So you need a loop to repeatedly print many $ signs on the same line.
- You also need to print many lines. That means the code that prints many characters on each line (which is a loop) will itself need to go inside another loop.
5.5 Printing triangles

Write a program, using only 2 `while` loops, to print the following triangle on the screen:

*  
**  
***  
****  
*****

Hint: Each line number has a relationship with the number of stars that need to be printed on that line. For example, on the 1st line, you need to print 1 star, and on the 2nd line, you need to print 2 stars. So that is how you can know how many stars to print, depending on which line you are currently on.

Please open *all* your exercises *before* calling the tutor to get marked off and leave early.