ENGGEN 131 Assignment 1: Matlab Test Preparation

This assignment is designed to help you prepare for your upcoming Matlab test.

Your test will contain four questions. One will be a debugging exercise and the other three will require you to write functions. This assignment has been structured in the same way. The level of difficulty of the questions in this assignment is in line with what you can expect for the test. Each question has been designed to take around 20 minutes to complete (assuming you have mastered the concepts that the questions test).

As with the test, some of these assignment questions are harder than others – this helps us identify those students who have truly mastered the material.

You will submit your assignment answers to Matlab Grader (you have 3 attempts for each function)

There are some key differences between this assignment and your test, namely:

This assignment doesn't cover all of chapters 1-9 inclusive, your test could cover any examinable material from the first 9 chapters. While the assignment will give you an idea of the kinds of questions we ask in tests, you shouldn't assume that the test will only be limited to concepts covered in this assignment (e.g. we might put a question in the test that asks you to do something with files or 3D arrays, neither of which are concepts covered in this assignment).

This assignment is marked by Matlab grader, whereas the test is marked by humans. An important consequence of this is that you can get part marks for test questions, even if your code wouldn't run (whereas code that doesn't run will earn you no marks on Matlab grader, as it won't pass any of the automated tests)

You have a large amount of time to complete this assignment, whereas your test is only 90 minutes long. You might like to simulate this being a timed assessment by trying to complete all four questions in a single 90-minute session. This will help you get a better sense of what doing the test will be like (but feel free to take longer than 90 minutes on this assignment if you need to).

You can type this assignment whereas you will handwrite your test (unless you are an online student). Because the in-person test rooms don't have computers, you will be handwriting your answers. Feel free to use a computer for this assignment however you might like to try and simulate your test conditions by first trying to write your answer without running it in Matlab, as you won't have access to Matlab during the test. We realise that it is harder to write good code when handwriting (as you don't get a chance to run your code) but we account for this when marking tests. Also during a test you don't have heaps of time to spare on debugging – the focus in a test or exam is answering as much of the question as you can, in the time available.

The assignment has no marks associated with style, whereas the test does. For the test we will be awarding style marks. When answering test questions you should include a header comment for your functions, unless the question tells you not to. Remember to include other comments and indent your code. Even though the assignment has no marks associated with style, I recommend still commenting and indenting your assignment code, as part of preparing for the test. The debugging question on the test may also ask you to identify problems with style.

You can look at your coursebook when completing this assignment, whereas for the test you can only bring in a single A4 sheet of notes. You might like to try and do this assignment without looking at your coursebook, to see how far you can get (and if you do need to look something up, consider adding it to your A4 sheet of notes).

For the assignment you need to find all the bugs in the debugging question, whereas the test will only ask you to identify a certain number. Because you are submitting your debugging question to Matlab grader you will need to fix ALL the bugs in order to receive full marks. For the test you will be asked to identify a certain number of bugs in some supplied code and will earn marks for each bug identified and fixed.

Remember that for the test you should attempt all questions as even a partial answer can earn you marks. One small error isn't going to cost you all your marks. Typically, if a question is worth 10 marks and you make a small error, you would only lose 1 of the 10 marks.

Question 1

A function `SwapIndicator` has been written that is to be used as part of a process of sorting numbers (where the eventual aim is to have items sorted in decreasing order). This function iterates through each pair of adjacent elements of a 1D array of numbers. For each pair, it records a true outcome if the right-hand element of the pair is higher in value than the left-hand element (indicating that the pair should be swapped), and a false value otherwise. Note that the function doesn't actually perform any swapping of elements.

The function will take one input:

1) `array`: a 1D array of numbers.

The function will return two outputs:

1) `is_swap`: a 1D **logical** array containing whether the pair of adjacent elements should be swapped.
2) `num_swaps`: the total number of swaps that should be performed.

An example call to the function would be:

```
array = [4, 5, 2, 3, 7, 8, 4]
[is_swap, num_swaps] = SwapIndicator(array)
```

This would return the following values for the outputs:

```
is_swap = [1, 0, 1, 1, 1, 0]
num_swaps = 4
```

Unfortunately, the function currently does not work as intended, due to numerous bugs. Find and fix the bugs so that the function works as intended.

```
function (is_swap, num_swaps) = SwapIndicator(array)
    is_swap = logical(zeros(length(array)-1));
    for i = 1:length(array)
        if array(i+1) == array(i)
            is_swap(i) = 0;
            num_swaps = num_swaps + 1;
        else
            is_swap(i) = 1;
        end
    end
end
```

Question 2

Write a function `GoBeach` that will help you to determine which of the coming days you should go to the beach. You should only go to the beach when it is not raining and also if the temperature is between 20 and 30 celsius, inclusive.

The function will take two inputs (taken from weather forecasts):

1) `temperature`: a 1D array containing the air temperature for the next N days.
2) `rain`: a 1D logical array containing if there will be rain (true) or not (false) for the next N days.

The function will return two outputs:

1) `is_beach`: a 1D **logical** array detailing whether you should go to the beach (true) or not (false) for the next N days.
2) `num_beach_days`: how many of the next N days you should go to the beach.

An example call to the function would be:

```
temperatures = [17, 23, 33, 25]

rain = [false, true, false, false]

[is_beach, num_beach_days] = GoBeach(temperatures, rain)
```

This would return the following values for the outputs:

```
is_beach = [0, 0, 0, 1]

num_beach_days = 1
```

Hint: You may find the logical function useful (it can be used to convert an array to type logical, e.g.

```
A = logical(A) % take an array and convert it into a logical array
```

Question 3

In the aviation industry every international airport has an abbreviation that is used to identify it. Each abbreviation consists of three capital letters, e.g. AKL identifies Auckland International Airport, whereas LAX identifies Los Angeles International Airport.

You have been tasked with writing a hashing function called `HashSquares` that will find hash values for three letter international airport codes. Your function will take as an input a three-letter character array (containing only uppercase letters) and return as an output a (hopefully unique) integer hash value, calculated by using the following algorithm:

1) Convert the three-letter string of uppercase characters into a 6-digit integer

   (We will write a helper function to do this task)

2) Square the 6-digit integer and then write the resulting 12-digit value out to a string
3) Extract the middle four characters from the 12-character string
4) Convert this string into an integer, this is your hash value.

Here is a worked example for the input 'AKL'

1) Convert the three-letter string of uppercase characters into a 6-digit integer

The character array 'AKL' has corresponding ASCII values [65 75 76].

The 6-digit integer associated with these values is 657576

2) Square the 6-digit integer and then write the resulting squared value out to a string

Squaring 657576 and writing it out to a string gives the string `'432406195776'`

3) Extract the middle four characters from the string

This string has 12 characters in it. The middle 4 characters are shown highlighted in red

`'432406195776'`

4) Convert this string into an integer, this is your hash value.

We would return the integer value 619

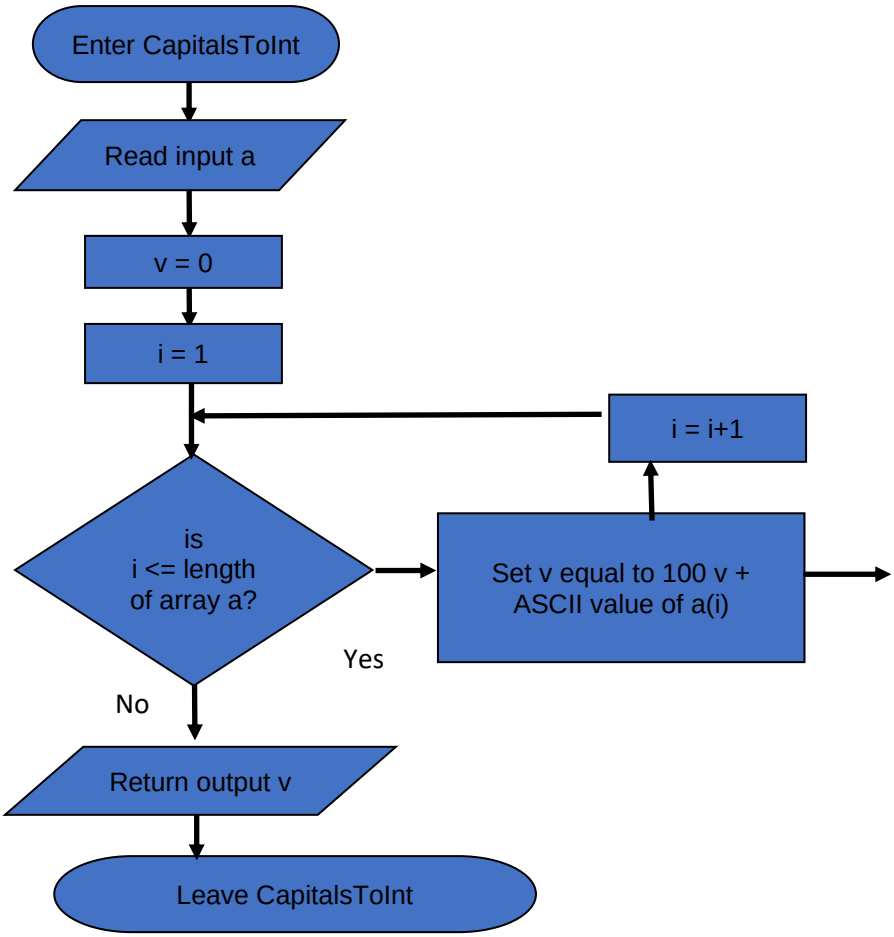
Here are some example calls:

```
>> h = HashSquares('AKL')  % will return 619
>> h = HashSquares('LAX')  % will return 5716
```

This question has two parts

1) Write the helper function `CapitalsToInt` which is described on the following page
2) Write the function `HashSquares` (which should use your `CapitalsToInt` function)

Below is a flow-chart that describes how to convert a string of capital letters store in a character array, a, into an integer value, v.



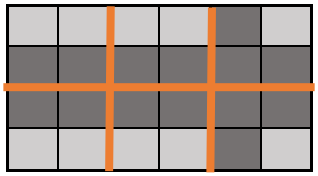
Here are some example calls to `CapitalsToInt`

```
>> v = CapitalsToInt('A')

v =

    65

>> v = CapitalsToInt('AK')

v =

        6575

>> v = CapitalsToInt('AKL')

v =

      657576

>> v = CapitalsToInt('LAX')

v =

      766588
```

Question 4

Write a function called `HalveImage` which will reduce a greyscale image to half it's width and height by using the following algorithm.

1) Check if our image is an even number of pixels wide, if not add a duplicate of the right most column of pixels to the right-hand side of the image (to ensure we have an even number of columns, c).
2) Check if our image is an even number of pixels high, if not add a duplicate of the bottom column of pixels to the bottom of the image (to ensure we have an even number of rows, r).
3) Divide the image up into a grid of 2x2 squares (i.e. each square contains four pixels, as in the small example image below).



Note that there will be r/2 rows and c/2 columns in the grid of 2x2 squares (e.g. if the original image contains 4 rows and 6 columns of pixels, the grid will contain 2 rows and 3 columns of 2x2 squares, as shown above).

4) Construct a new image by finding the average value for each 2x2 square of pixels and placing a single pixel with this average value into the corresponding position in the new image. E.g. the pixel that corresponds to the first 2x2 square (in row 1, column 1 of the grid of 2x2 squares) will have a value assigned to the pixel in row 1, column 1 of our new image. This value will be the average (i.e. mean) of the four values contained within the first 2x2 square.

Your function should take in 1 input, a 2D array of `uint8` values representing a greyscale image.

Your function should return one output, a 2D array of `uint8` values containing a greyscale image of half the size.

Hint: you may find the `mod` function helpful in determining if a row or column has an even number of pixels in it (as if you divide a number by 2 and the remainder is zero, you know the n
umber must be even).