Assignment 10

Matlab Assignment

Date Due: Tuesday, April 13, 11:59pm **Total Marks: 28**

General Instructions

- **This assignment is individual work.** You may discuss questions and problems with anyone, but the work you hand in for this assignment must be your own work.
- Each question indicates what to hand in. You must give your document the name we prescribe for each question, usually in the form **aNqM**, meaning Assignment N, Question M. **Put your name, NSID, student number and instructor's name at the top of every document you hand in.** These conventions assist the markers in their work. Failure to follow these conventions will result in needless effort by the markers, and a deduction of grades for you. Do not submit folders, zip documents, even if you think it will help.
- **Programs must be written in Matlab** Marks will be deducted with severity if you submit code with any format other than .m.
- **Assignments must be submitted to Moodle.** There is a link on the course webpage that shows you how to do this.
- **Moodle will not let you submit work after the assignment deadline.** It is advisable to hand in each answer that you are happy with as you go. You can always revise and resubmit as many times as you like before the deadline; only your most recent submission will be graded.
- Questions are annotated use descriptors like "easy" "moderate" and "tricky". All students should be able to obtain perfect grades on "easy" problems. Most students should obtain perfect grades on "moderate" problems. The problems marked "tricky" may require significantly more time, and only the top students should expect to get perfect grades on these. We use these annotations to help students be aware of the differences, and also to help students allocate their time wisely. Partial credit will be given as appropriate, so hand in all attempts.

Matlab Instructions

Matlab is available via the University of Saskatchewan **Virtual Computer Lab**:

- Open **https://vlab.usask.ca** in a web browser. Log in using your USASK NSID and password.
- From the grey sidebar (on the left) select "Common U of S". Followed by "MATLAB 2019a Common".

Access your **files** through **Cabinet**:

- log in to **https://webshare.usask.ca:4433/** via a web browser
- **upload** or **download** files (feel free to create appropriate **folders**)
- access files in **MatLab** under the **V:** drive

Question 1 (8 points):

Purpose: To practice using I/O and loops in Matlab

Degree of Difficulty: Easy

Below is a code originally written in Python 3.

```python
import random as r

secret = r.randint(1, 10)

guess = int(input("Guess a number from 1 to 10! "))
while guess != secret:
    if guess < secret:
        print("Try higher!")
    else:
        print("Try lower!")
    guess = int(input("Guess again: "))

print("You got it, the secret number was", secret)
```

Your task is to create a Matlab implementation of this solution.

Hints

- you can use the function `randi([a, b]);` to generate a random integer between a and b (inclusive).
- you can use `fprintf()` or `sprintf()` to do "fancy" printing in Matlab.

What to Hand In

- A document called `a10q1.m` containing your code as described above.

Be sure to include your name, NSID, student number and instructor's name at the top of all files.

Evaluation

- 1 mark for correctly generating a random number
- 1 mark for getting user input
- 1 mark for the while loop
- 2 marks for the if statement
- 1 mark for displaying the correct information to the console
- 2 mark for documentation (comments)
- **-1 mark: for missing name, NSID and student number at top of file**

Question 2 (5 points):

Purpose: To practice using the MatLab interface, practice with Matrix multiplication.

Degree of Difficulty: Easy

Dick Gumshoe is an underpaid Detective. As a result, he only buys the cheapest of foods. Below is a chart showing the individual cost (in Candian dollars) of a food item Gumshoe may buy.

```
Mac and Cheese:      0.99
Cup-a-Soup:          1.37
Chunky Chili:        1.85
```

Below is a chart showing the food purchases Detective Gumshoe has made in the last 6 months:

```
                    Sept  Oct   Nov   Dec   Jan   Feb
Mac and Cheese      9     12    13    7     6     21
Cup-a-Soup          6     31    5     9     21    13
Chunky Chili        19    14    3     6     9     12
```

Use MatLab to create a program (with a .m extension) that calculates the cost of food for Gumshoe each month. This should be done using matrix multiplication **(remember, every variable in MatLab IS a matrix!)**

You should need no more than 3 lines of code. You will not need any loops.

Your file will need to output the cost of food for the past 6 months in a single line to the console when run. Your output should look like the following:

```
costsPerMonth =

     52.2800    80.2500    25.2700    30.3600    51.3600    60.8000
```

Hints:

- The order of operands may matter in matrix multiplication. If you're getting errors, try swapping the position of the operands
- If you don't put a semi-colon at the end of a line, that line will be printed as output to the console.

What to Hand In

- A document called `a10q2.m` containing your code as described above.

Be sure to include your name, NSID, student number and instructor's name at the top of all files.

Evaluation

- 3 marks: Your file output matches the output described above.
- 2 marks: Your file correctly performs matrix multiplication.
- **-2 marks: for using loops**
- **-1 mark: for missing name, NSID and student number at top of file**

Question 3 (5 points):

Purpose: To practice using the MatLab interface, practice with Matrix Arithmetic Operation.

Degree of Difficulty: Easy

After the fourth shinobi world war, it becomes a rule in Konoha that to be a ***Genin*** (the lowest level of ninja) from an academic student (ninja-student) anyone needs to get 1000 points. A student can earn points by finishing different missions. Below is a chart showing the individual mission points based on the difficulty level:

```
X-rank:   48
Y-rank:   32
Z-rank:   76
```

Below is a table showing the number of missions completed by Sarada, Boruto, Shikadai, and Mitsuki in the last academic year:

	Sarada	Boruto	Shikadai	Mitsuki
X-rank	9	5	11	8
Y-rank	5	12	9	7
Z-rank	7	3	6	8

Your job is to use MatLab to create a MatLab file (with a .m extension) that calculates the total points that they earned in an academic year. This should be done using matrix multiplication (remember, every variable in MatLab is a matrix!)

Also, you need to show the Genin status of Sarada, Boruto, Shikadai, and Mitsuki by comparing the total points with the required pass point "1000". This should be done by creating a row vector which only consists 0 and 1 (where 1 indicates pass and 0 indicates fail).

You should need no more than 4 lines of code. You will not need any loops.

Your file will need to output the Genin status of Sarada, Boruto, Shikadai, and Mitsuki in a single line to the console when run. Your output should look like the following: ***GeninStatus* = 1 0 1 1**

Hint 1: The order of operands may matter in matrix multiplication. If you're getting errors, try swapping the position of the operands.

Hint 2: If you don't put a semi-colon at the end of a line, that line will be printed as output to the console.

This isn't a trick question! This question is to get you in the mindset of using Matrix math.

What to Hand In

- A document called `a9q1.m` containing your code as described above.

Be sure to include your name, NSID, student number, course number and lecture section and laboratory section at the top of all documents. Comments in MatLab are denoted with the % symbol.

Evaluation

- 1 mark: Your file is correctly named `a10q3.m`.
- 2 mark: Your file output matches the output described above.
- 1 mark: Your file correctly performs matrix multiplication.
- 1 mark: Your file correctly performs row vector calculation which holds logical values.

Question 4 (10 points):

Purpose: Writing functions in MatLab, data processing

Degree of Difficulty: Moderate

Acme Engineering Company (AEC) specializes in construction of large concrete structures. They use live sensors to monitor the temperature of concrete during curing. Recently, some projects have gone poorly, and **AEC** suspects that some of their sensors may be faulty. In this question, we are going to read a log of a single sensor's data and determine if the sensor is faulty using **MatLab**.

A file named **concrete.txt** has been provided which contains the concrete sensor's data. Download it and place it in the **same folder** as your **a10q4.m** file.

Program Design

Write code to read the data from **concrete.txt** into a **table** using the **readtable** function.

Below is an example of reading data from a file:

```
table_var = readtable("filename.txt");
```

tables are similar to matrices in **MatLab** but differ in the following ways. They can have different types of values in their columns, instead of an ordinary matrix whose values must all be of the same **type**.

To **access values** inside an entry of our table, use { } **curly brackets**. Accessing the first item of a table would be like this example:

```
table_var{1, 1};
```

Sensors are expected to record incomplete or invalid data occasionally. However, if this happens on more than 10% of the time, **AEC** deems the sensors to be **faulty**. Invalid entries are denoted as **"NaN"** which means **"Not a Number"**. In **MatLab** you can check if a value is **NaN** with the function **isnan**. For Example:

```
isnan(value);
```

Create a function **testsensor** which takes a **table** as a parameter and returns the **number of invalid entries** as well as a **vector** of invalid entry **row numbers**. Remember **MatLab** functions can return multiple values!

An example of a function with multiple return values:

```
function [return1 return 2 return3] = somefunction(input1, input2)
    % some code
end
```

An example of calling a function and storing multiple return values:

```
[var1 var2 var3] = somefunction(input1, input2)
```

Inside the **testsensor** function, you will need to:

- Loop through every row in the **table**
- Check to see if each row's sensor data **is NaN**
- Create a **vector** of rows which have **NaN** values
- Return a count of the number of invalid entries **and** a **vector** containing the rows which have **NaN** values.

Hint: To loop through rows in a table you can use a for loop. Below is an example of looping through rows in a table and printing the first **column** in that row:

```
for row_index = 1:height(table)
    table{row_index,1}
end
```

After running your **testsensor** function with the table created from the **concrete.txt** file, **print** your results to the console. You should include the number of entries that are **faulty**, the **vector** of **faulty** rows, and say if the sensor is **faulty** (above the 10% threshold). For example:

```
Out of 50 sensor logs we found 7 invalid entries.
Invalid data was contained on rows:
    10    16    17    19    26    27    34
This sensor appears to be faulty.
```

What to Hand In

- A document called `a10q4.m` containing your code that calls the function **testsensor** as described above.

Be sure to include your name, NSID, student number and instructor's name at the top of all files.

Evaluation

- 2 mark: Your file correctly reads in `concrete.txt`
- 2 marks: Your function `testsensor` is correctly defined, with the proper parameter, and return values.
- 2 mark: `testsensor`'s docstring is sufficient.
- 1 mark: `testsensor` properly counts the number of rows containing NaN.
- 1 mark: `testsensor` properly stores each row index containing NaN.
- 2 mark: Your program calls your function `testsensor` and then prints an appropriate message.
- **-1 mark: for missing name, NSID and student number at top of file**