GEOS397

Homework 1

Part 1:

Using my Boise State University student email address, I created a GitHub account. From here, I cloned the *master* branch of GEOS397 to my local directory on my laptop. I created a new branch called "GEOS397_Dudunake".

Part 2:

After creating my own branch, I created a new directory for Homework 1 where this .md file is located on my local directory. Next, I've been tasked with explaining how I would partner with everyone in a class of 10 total students for each of the 9 total homework assignments. The only caveat is that no two students can have repeat partners.

This first logical thing I thought of was my prior knowledge of round-robin tournaments. For the case of this problem, $N = 10$ students. To figure out how many total assignments will be needed so that everyone is partners with everyone for the 9 assignments, we can use the equation $N/2 * (N - 1)$. By using this equation, there will be a total of 45 assignments turned in at the end of the semester. To set this up, I have created a set of 9 scenarios representing each of the 9 assignments.

Column1	Group 1	Group 2	Group 3	Group 4	Group 5
Hwk 1	1	2	3	4	5
	10	9	8	7	6
Hwk 2	1	10	2	3	4
	9	8	7	6	5
Hwk 3	1	9	10	2	3
	8	7	6	5	4
Hwk 4	1	8	9	10	2
	7	6	5	4	3
Hwk 5	1	7	8	9	10
	6	5	4	3	2
Hwk 6	1	6	7	8	9
	5	4	3	2	10
Hwk 7	1	5	6	7	8
	4	3	2	10	9
Hwk 8	1	4	5	6	7
	3	2	10	9	8
Hwk 9	1	3	4	5	6
	2	10	9	8	7

As you can see, I essentially assigned each student a number 1 through 10 then laid out each number as seen in the "Hwk 1" row. Then, keeping student 1 in the same position I rotated student 2 through 10 in a clockwise fashion throughout the table. This ensures that each student is partnered with a different person on each homework assignment. Also, no two students are paired with a repeat partner. For example, on HWK1, student 1 is paired with 10, student 2 is paired with 9, student 3 is paired with 8, student 4 is paired with 7, and student 5 is paired with 6. On homework 2, all the partners change as seen in the above graph and continues to change through the rest of the assignments through the semester.

Part 3:

Below is a list of all the possible variable types found in MATLAB as explained in the style guide reading assignment. It's important to note that variables should be mixed case and begin with a lower case character. Also, variable names are sensitive to upper and lower case.

Floating-point numbers: These exist as *single* and *double* class precision and is the default data type in Matlab. This data type is useful when storing real numbers. Example: A = 5.83

Integers: These exist in many different classes such as int8, uint8, int16, uint16, etc. This data type is useful when storing whole numbers and is more memory efficient. They exist in 8, 16, 32, and 64 bits of storage. Example: A = uint32(50)

String and characters: These exist as *char* classes. They are useful when storing text. A string may be an array of characters while *char* signifies each character. Example: A = 'Taylor'

Boolean: These exist as a *logical* class. They are useful when storing 1 and 0, true and false, respectively. Example: A = true

Structures: These exist as a *struct* class. They are useful when storying varying types and sizes of data. These data are accessed using a name. Example: A = struc('Name')

Cell Arrays: These exist as a *cell* class. They are useful when storying varying lengths and types of data. The data within the arrays are accessed using an index. Example: A = cell(21)

Part 4:

Examples of each variable type:

- Floating-Point numbers
  - A = 1.25
  - x = 5.73 * 10^300
- Integers
  - x = int32(50)
  - b = uint16(25)
- String and characters
  - x = 'Taylor'
  - y = 'John'
- Boolean
  - 0 or 1
- Structures
  - field = 'f'; value = {'Win';[10, 20, 30]}
- Cell Arrays
  - c = {'one', 'two', 'three'; 1, 2, 3}