lab04_HW

1. Write a function that takes a matrix as its input. The entries of the matrix are non-negative integers of type double. The function returns the variable type (e.g., uint8, uint16, uint32, and uint64) of the unsigned integer class to which the matrix can be accurately converted. For example, if the largest integer in the matrix is 14, the function returns uint8. If no such class exists, the string ‘NONE’ is returned.

This question is just let you know the range of int8 …

see `typeMin.m`

```
>> typeMin(double([2^3 2^8-1]))
ans =
    'uint8'
>> typeMin(double([2^3 2^8]))
ans =
    'uint16'
>> typeMin(double([2^3 -1]))
ans =
    'None'
```

2. Write a script that generates a cell array of dimensions 31-by-3. The rows of the array correspond to the days of October, 2015. The three elements of each row must be set as follows:
   - The first element refers to the string ‘October’ (uppercase ‘O’).
   - The second element refers to a scalar of type double that equals the date (1 through 31).
   - The third element refers to the three-letter abbreviation of the day chosen from this list: `'Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', and 'Sun'` .

     The eleventh element of the cell array should be:

     `'October' [11] 'Sun'.`

By reading doc. We find cat can combine cell array (also matrix)

see `OctoberGenerate.m`

```
>> OctoberGenerate
    'October'    [ 1]    'Thu'
    'October'    [ 2]    'Fri'
    'October'    [ 3]    'Sat'
    ...
    ```
```

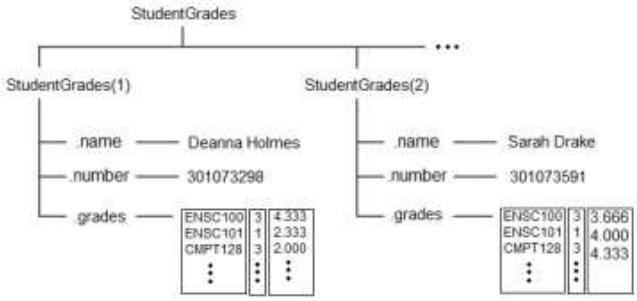
3. Write a function that takes a string representing an integer between 1 and 20 inclusive using Roman numerals and returns the Arabic equivalent as a number of type uint8. If the input is illegal or its value is larger than 20, the function returns 0 instead. Allow only three consecutive symbols can be the same in your function. For example, IIII or VIIII are illegal.

Just logic for if and for

see `toRoman.m`

```
toRoman(19)
ans =
    'XIX'
```

4. Write a script that reads a data structure StudentGrades from “04Student_grades.mat”. The structure is organized as the follows:



The fields are:

- name – a string specifying the student’s name.
- number – a scalar specifying the student’s number.
- grades – a 1-by-3 cell array where:
  - The 1st cell is a 9-by-7 char matrix specifying the course names.
  - The 2nd cell is a 9-by-1 vector with the corresponding credits for each course of the 1st cell.
  - The 3rd cell is a 9-by-1 vector with the corresponding grade that the student earned for
```

each course of the 1st cell. These are in the form of a 4.333 gradescale (e.g., 4.333 is an A+, 4.000 is an A, 3.666 is an A-, 3.333 is a B+, etc.).
The script should calculate each student's cumulative GPA (CGPA) for a set of N courses, $$\mathrm{CGPA} = \frac{\sum_{i=1}^{N} c_i g_i}{\sum_{i=1}^{N} c_i},$$

where g_i and c_i are the grade number of credits for the ith course. The script should write the students' names, numbers, and CGPAs into an Excel XLSX file. Include the header in the file.

Excel is evil application, so I use `csv` file

Further more, matlab don't have concatenateStruct, so I write simple one

Note that matlab does not have struct2csv.

In constract, use convert2table and writetable is another method.

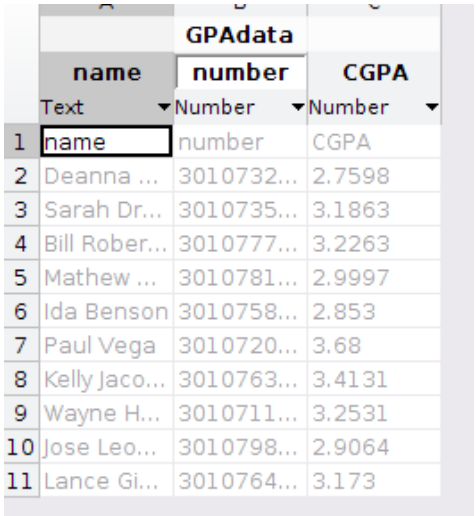
see `gpaCalc.m`

see `GPAdata.csv`

```
>> gpaCalc('04Student_grades.mat','GPAdata.csv')
          name              number        CGPA
    ________________    __________    ______
    'Deanna Holmes'     3.0107e+08    2.7598
    'Sarah Drake'       3.0107e+08    3.1863
    'Bill Roberson'     3.0108e+08    3.2263
    'Mathew Klein'      3.0108e+08    2.9997
    'Ida Benson'        3.0108e+08     2.853
    'Paul Vega'         3.0107e+08      3.68
    'Kelly Jacobs'      3.0108e+08    3.4131
    'Wayne Harvey'      3.0107e+08    3.2531
    'Jose Leonard'      3.0108e+08    2.9064
    'Lance Gibbs'       3.0108e+08     3.173
```

wiil create `GPAdata.csv`

looks like



GPAdata

	name	number	CGPA
	Text	Number	Number
1	name	number	CGPA
2	Deanna ...	3010732...	2.7598
3	Sarah Dr...	3010735...	3.1863
4	Bill Rober...	3010777...	3.2263
5	Mathew ...	3010781...	2.9997
6	Ida Benson	3010758...	2.853
7	Paul Vega	3010720...	3.68
8	Kelly Jaco...	3010763...	3.4131
9	Wayne H...	3010711...	3.2531
10	Jose Leo...	3010798...	2.9064
11	Lance Gi...	3010764...	3.173

5. Write a script to generate a table of the logarithm values of base ten, i.e., y = log 10 x. The x ranges between 1 and 10 in steps of 0.1. The results should be arranged in a table as shown below. The table should include a title in the first row, and row and column headings. The script should store the table into an Excel XLSX file.

Excel is evil application, so I use `csv` file

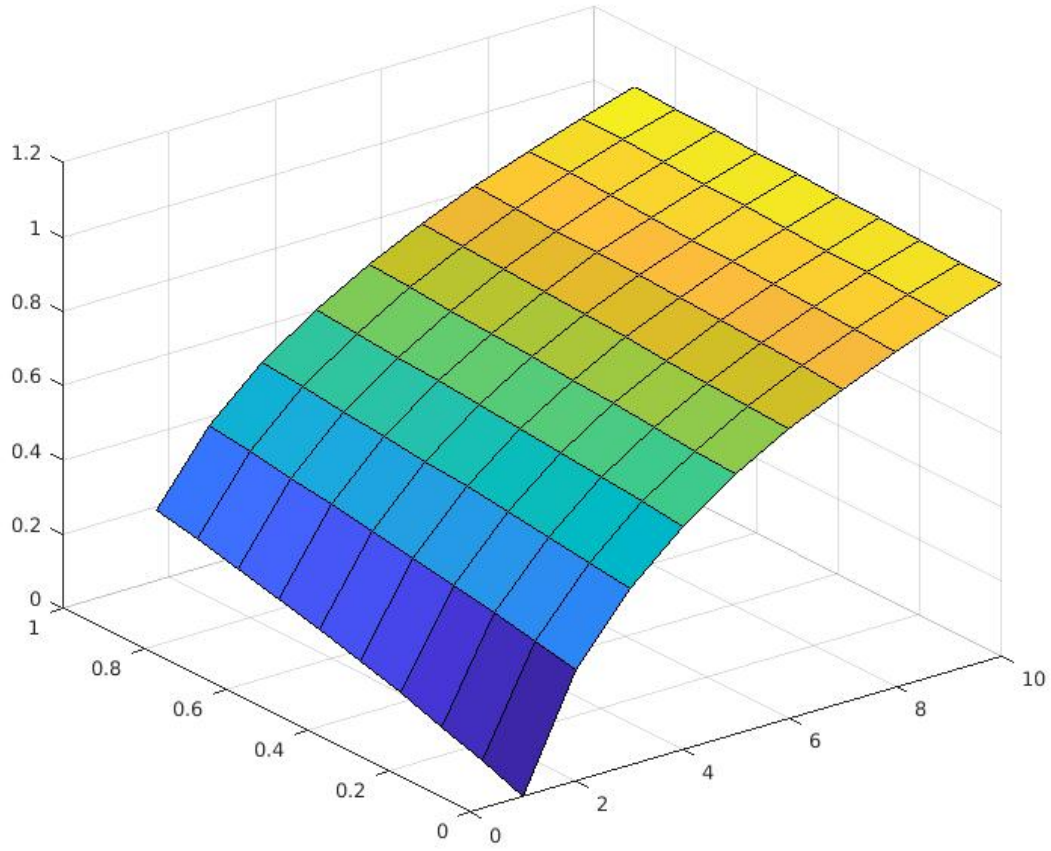
be care for table column name may be invalid, I rewrite csv file after table write into it.

see `log10generate.m`

see `log10.csv`

first use meshgrid to produce xy

```
>> [y, x] = meshgrid(linspace(0, 0.9, 10), linspace(1, 10, 10));
>> surf(x, y, log10(x + y));
```



second, save to csv

```
>> log10generate('log10.csv')
            x0          x1          x2          x3          x4          x5
          _____    ________    ________    _______    _______    _____

    1           0    0.041393    0.079181    0.11394    0.14613    0.176
    2     0.30103     0.32222     0.34242    0.36173    0.38021    0.397
    3     0.47712     0.49136     0.50515    0.51851    0.53148    0.544
    4     0.60206     0.61278     0.62325    0.63347    0.64345    0.653
    5     0.69897     0.70757       0.716    0.72428    0.73239    0.740
    6     0.77815     0.78533     0.79239    0.79934    0.80618    0.812
    7      0.8451     0.85126     0.85733    0.86332    0.86923    0.875
    8     0.90309     0.90849     0.91381    0.91908    0.92428    0.929
    9     0.95424     0.95904     0.96379    0.96848    0.97313    0.977
    10          1      1.0043      1.0086     1.0128      1.017     1.02
```

I rewrite variable name after table write it.(Bad method)

6. Download the file “04Stocks.txt” from the course website. Write a script to open the file and read all the data to memory. Your script should sort the data by the 3 rd column in ascending order. Your script should then save sorted data to a new file “Stocks_sorted.txt”.

Easy

see `stockSort.m`

see `Stocks_sorted.txt`

```
>> stockSort('04Stocks.txt','Stocks_sorted.txt')
ans =
  8×3 table
     Var1       Var2      Var3
    _______    _____    _____
    'WAMUQ'     0.07     0.06
    'WB'        5.31      5.8
    'C'        13.25    13.55
    'MS'          15    15.64
    'BAC'      20.49    21.07
    'WFC'      29.04    30.91
    'JPG'      35.12    35.43
    'GS'       93.35    98.34
```

Test on ubuntu16.04 + Matlab R0217a academic use