VG101 Lab Worksheet

Lab 3

Instructor: Dr. Yifei ZHU
TA: Hangrui CAO
TA: Qinhang WU
TA: Muchen XU

Code Quality Requirement

- Ensure proper indentation
- Make naming meaningful
- Include necessary comments
- Split the code over functions
- Test the code as much as you can

Category	Exercise title	Tag
basic syntax exercise	/	function, basic plot
practical exercise	Narcissistic Number	loop, function
practical exercise	Goldbach's Conjecture	function, enumeration
practical exercise	Minesweeper	function
practical exercise	Pascal Triangle	formatted output, FileIO
practical exercise	Complex Electric Field Simulation	3D plot, MVC

- In-lab quiz will test your understanding of some of these problems, get prepared.
- Remember to submit your code on time, otherwise your attendance score will be deducted.
- Coding is not that difficult. Think before programming to see whether there is an easier implementation. Make your life easier.

Basic Syntax Exercise

Function

In this part, you may use the code skeleton provided below to fulfill the following requirements. You're free to specify anything by yourself that is not covered in the problem statement.

1. Complete the function `isSquared` that returns 1 if the parameter is a **perfect square**. Otherwise, return 0.
2. Write a function that returns the **surface area** and **volume** with height and radius parameter given.
3. Complete the function `allDigits` that returns all digits of a parameter as a vector.
4. Complete the function `isPrime` that returns 1 if the integer parameter is a **prime number**. Otherwise, return 0. You're not allowed to use the MATLAB function `isprime`.

Code skeleton you may use:

```
1  clearvars;clc;
2  disp([isSquared(257) isSquared(256)])
3  [a,v] = calcCylinder(1,3);
4  disp([a,v])
5  disp(allDigits(1234))
6  disp([isPrime(1) isPrime(23333) isPrime(19260817)])
```

```
 7
 8   %Don't modify the code above
 9
10   function result=isSquared(x)
11        % TODO: check whether x is a square number
12   end
13
14   %-----------------------------------------------------------%
15   % TODO: Define and finish the calcCylinder function here %
16   %-----------------------------------------------------------%
17
18   function result=allDigits(x)
19        % TODO: return all the digits of x, can be done in one line
20   end
21
22   function result=isPrime(x)
23        % TODO: check whether x is prime
24   end
```

Note that the input has already been included in the code skeleton. Expected output is:

```
1         0     1
2
3      75.3982    28.2743
4
5         1     2     3     4
6
7         0     1     1
```

Plot

5. Use `plot` to draw functions $y_1 = sin(x)$ and $y_2 = cos(x)$ **at the same time**.
6. Use `fill` to draw several **polygons** like squares, rectangle, octagon, circle **at the same time**.
7. Animate a clockwise-rotating square and a counter-clockwise-rotating equilateral triangle at the same time.
8. Wrap problem 5 and 6 into functions.

Practical Exercises

Narcissistic Number

Narcissistic number is a number that is the sum of its own digits each raised to the power of the number of digits, like $9474 = 9^4 + 4^4 + 7^4 + 4^4$

Use the function implemented in 1.3 to find all the Narcissistic number smaller than 10^8.

Goldbach's Conjecture

Use the function implemented in 1.4 to validate the Goldbach's Conjecture: Input an even number n, find two prime number p, q that $p + q = n$.

Sample input/output:

```
1  9870
2             11         9859
```

Minesweeper

Write a function `myMinesweeper(r, c)`, which takes the number of rows `r > 3` and columns `c > 3` as inputs, and randomly generate an r x c truth board, with 0 indicating safe position, and 1 indicating a mine in that position.

Start from an empty board containing only 0s. Then until the user wins (with all mines identified without touching them) or loses (touched a mine), the user is prompted to input two numbers `[r,c]` indicating a guess **(comma separated, bracket enclosed)**. Then the number in the position is updated with the number of mines surrounded, i.e., the number of mines within a 3 x 3 grid centered at the guessed location. If the game ends, the user should be prompted with either `Congratulations, you wins`, or `Sorry, you lose`. For each intermediate round, the updated board should be printed in the command window.

```
1   4
2   4
3   [2 2]
4        0      0      0      0
5        0      5      0      0
6        0      0      0      0
7        0      0      0      0
8
9   [4 4]
10       0      0      0      0
11       0      5      0      0
12       0      0      0      0
13       0      0      0      2
14
15  [2 4]
16       0      0      0      0
17       0      5      0      5
18       0      0      0      0
19       0      0      0      2
20
21  [4 3]
22       0      0      0      0
23       0      5      0      5
24       0      0      0      0
25       0      0      3      2
26
27  [4 1]
28       0      0      0      0
29       0      5      0      5
30       0      0      0      0
31       1      0      3      2
32
33  [3 1]
34       0      0      0      0
35       0      5      0      5
36       1      0      0      0
```

```
37        1     0     3     2
38
39  [2 1]
40        0     0     0     0
41        2     5     0     5
42        1     0     0     0
43        1     0     3     2
44
45  [3 2]
46        0     0     0     0
47        2     5     0     5
48        1    -1     0     0
49        1     0     3     2
50
51  Sorry, you lose
```

Pascal Triangle

Pascal's triangle is a triangular array composed of the binomial coefficients. Write a MATLAB function taking as input an integer n and which outputs n lines of Pascal's triangle in a **text file**.

(Hint: either generate it using the fact that each number in the triangle is the sum of the two numbers directly above it or using the functions `pascal diag`, and `rot90`.)

Sample input/output:

```
1  7
2                  1
3               1     1
4            1     2     1
5         1     3     3     1
6      1     4     6     4     1
7   1     5    10    10     5     1
8  1   6    15    20    15    6    1
```

Complex Electric Field Simulation *

Try to simulate a charged particle's movement in an electric field formed by several fixed charges. You can modify the code in "Three-body Animation" part.

Can you extend the case to three dimension?