VG101 — Introduction to Computer and Programming

Lab 1

Instructor: Manuel

TA: Zhengyuan — UM-JI (Summer 2019)

Goals of the lab

- Basic scripting in MATLAB
- Control and loop statements
- File I/O operations
- C-style print function `fprintf`
- Basic functions of the Terminal

1 Introduction

This time Haruka, Kana, and Chiaki are interested in some Ancient Chinese Mathematics. Recently, they are attracted by Yang Hui, a great Chinese mathematician in Song dynasty. Yang Hui is is best known for his contribution of presenting Yang Hui's Triangle, which is also called Pascal's Triangle. After researching more about Yang Hui, Haruka, Kana, and Chiaki found that Yang Hui also developed some methods of constructing magic squares, which is the basis of the `magic` function in MATLAB. They want to discover more about magic squares.

Their friend Jane, who has a nickname "encyclopaedia", knows all about magic squares. He explained:

Magic Squares

A magic square is a $n \times n$ square grid (where n is the number of cells on each side) filled with distinct positive integers in the range $1, 2, \ldots, n^2$ such that each cell contains a different integer and the sum of the integers in each row, column and diagonal is equal. The sum is called the magic constant or magic sum of the magic square. A square grid with n cells on each side is said to have order n.

According to Jane's explanation, Haruka drew some magic squares taken from Yang Hui's treatise, as shown in Figure 1.

4	9	2
3	5	7
8	1	6

2	16	13	3
11	5	8	10
7	9	12	6
14	4	1	15

1	23	16	4	21
15	14	7	18	11
24	17	13	9	2
20	8	19	12	6
5	3	10	22	25

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

31	76	13	36	81	18	29	74	11
22	40	58	27	45	63	20	38	56
67	**4**	49	72	**9**	54	65	**2**	47
30	75	12	32	77	14	34	79	16
21	39	57	23	41	59	25	43	61
66	**3**	48	68	**5**	50	70	**7**	52
35	80	17	28	73	10	33	78	15
26	44	62	19	37	55	24	42	60
71	**8**	53	64	**1**	46	69	**6**	51

Figure 1: Magic squares of orders 3, 4, 5, 6, and 9 taken from Yang Hui's treatise.

Kana thought it will make things easier if they can write some script to draw the magic squares instead of hand drawing. He convinced Haruka so that they are planning to write a MATLAB script, which can

i) Get a input value n from a file.

ii) Print a magic square of order n.

iii) Save the result to another file.

However, they have forgot everything they learnt in the course VG101, so they don't know much about file I/O and formatted printing in MATLAB. After trying to find some useful information on Baidu but failed, they turned to you, experts in MATLAB, for help.

2 Work flow

2.1 Constructing a magic square

After asking Jane for more information, she said:

Construction of Magic Squares

There are many ways to construct magic squares, but the standard (and most simple) way is to follow certain configurations/formulas which generate regular patterns. Magic squares exist for all values of n, with only one exception: it is impossible to construct a magic square of order 2. Magic squares can be classified into three types: odd, doubly even (n divisible by four) and singly even (n even, but not divisible by four).

Haruka decided to write three functions to generate those types of magic squares. He divided you into three sections and each help him complete one of the function.

Simon will also provide your the detailed construction method of the type you are working on.

2.1.1 Generating Magic Square using MatLab Function (Mandatory)

Sometimes life can be so easy. When you find it hard to do something by your own effort, remember that MatLab has many installed functions that can satisfy you requirement very well, and what you need to do is simply looking up the documentation and learn how to use the functions correctly.

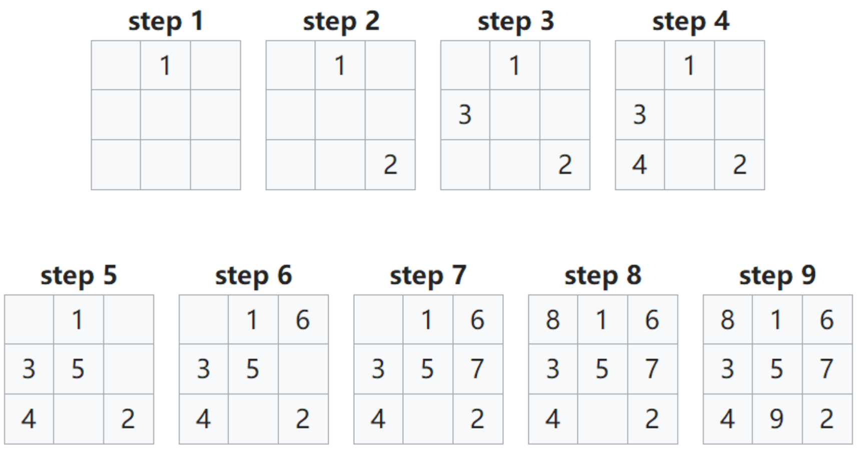
Please search for "magic square" using the MatLab documentation and generate a magic square of any given order.

2.1.2 Magic Square of Odd Order (Mandatory)

Write a function `mymagicodd(n)`, which takes an odd integer $n > 0$ as input, and return a magic square of order n as a matrix similar to `magic(n)`.

A method for constructing magic squares of odd order was published by the French diplomat de la Loubère in his book, A new historical relation of the kingdom of Siam (Du Royaume de Siam, 1693), in the chapter entitled *The problem of the magical square according to the Indians*. The method operates as follows:

The method prescribes starting in the central column of the first row with the number 1. After that, the fundamental movement for filling the squares is diagonally up and right, one step at a time. If a filled square is encountered, one moves vertically down one square instead, then continues as before. When an "up and to the right" move would leave the square, it is wrapped around to the last row or first column, respectively.



step 1

	1	

step 2

	1	
		2

step 3

	1	
3		
		2

step 4

	1	
3		
4		2

step 5

	1	
3	5	
4		2

step 6

	1	6
3	5	
4		2

step 7

	1	6
3	5	7
4		2

step 8

8	1	6
3	5	7
4		2

step 9

8	1	6
3	5	7
4	9	2

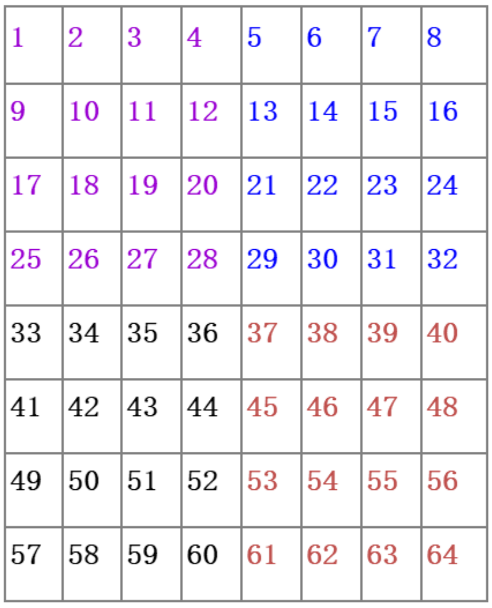
Figure 2: Construction of Magic Square of Odd Order.

2.1.3 Magic Square of Doubly Even Order (Optional)

Write a function `mymagicdoublyeven(n)`, which takes an doubly even integer $n > 2$ as input, , and return a magic square of order n as a matrix similar to `magic(n)`.

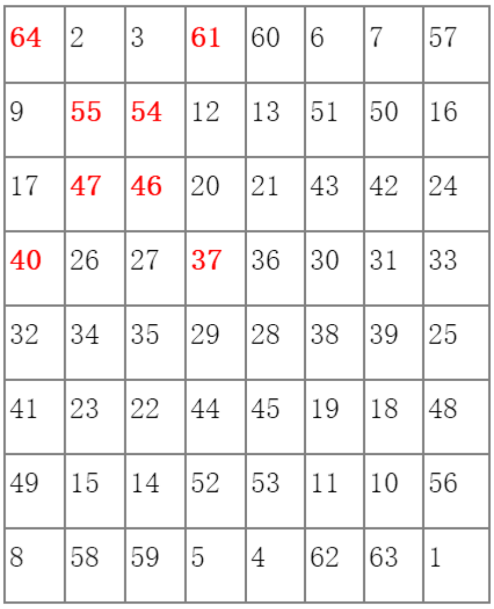
Doubly even means that n is an even multiple of an even integer; or $4k$ (e.g. 4, 8, 12), where k is an integer. Here is an example of Narayana-De la Hire's method.

All the numbers are written in order from left to right across each row in turn, starting from the top left hand corner. The square is split into $n^2/16$ parts of 4x4 small squares. Numbers in the small squares are then retained in the same place if they are not on the diagonal line, or interchanged with their diametrically opposite numbers if they are on the diagonal line. Two numbers a, b are diametrically opposite means that $a + b = n^2 + 1$.



1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

(a) Initialization.



64	2	3	61	60	6	7	57
9	55	54	12	13	51	50	16
17	47	46	20	21	43	42	24
40	26	27	37	36	30	31	33
32	34	35	29	28	38	39	25
41	23	22	44	45	19	18	48
49	15	14	52	53	11	10	56
8	58	59	5	4	62	63	1

(b) Result.

Figure 3: Magic square of order 8

2.1.4 Magic Square of Singly Even Order (Optional)

Write a function `mymagicsinglyeven(n)`, which takes an singly even integer $n > 2$ as input, , and return a magic square of order n as a matrix similar to `magic(n)`.

Conway's LUX method for magic squares is an algorithm by John Horton Conway for creating magic squares of order $4k + 2$, where k is a natural number.

Start by creating a $(2k + 1)$-by-$(2k + 1)$ square array consisting of

- $k + 1$ rows of Ls
- 1 row of Us
- $k - 1$ rows of Xs

and then exchange the U in the middle with the L above it, as shown in Figure 4.

Each letter represents a 2x2 block of numbers in the finished square.

Now replace each letter by four consecutive numbers, starting with 1, 2, 3, 4 in the centre square of the top row, and moving from block to block in the manner of the Siamese method (Section 2.1.2): move up and right, wrapping around the edges, and move down whenever you are obstructed. Fill each 2x2 block according to the order prescribed by the letter:

$$\text{L}: \begin{tikzcd} 4 & 1 \arrow[dl] \\ 2 \arrow[r] & 3 \end{tikzcd} \qquad \text{U}: \begin{tikzcd} 1 \arrow[d] & 4 \\ 2 \arrow[r] & 3 \arrow[u] \end{tikzcd} \qquad \text{X}: \begin{tikzcd} 1 \arrow[dr] & 4 \\ 3 \arrow[ur] & 2 \arrow[l] \end{tikzcd}$$

For example, let $k = 2$, so that the array is 5x5 and the final square is 10x10. Start with the L in the middle of the top row, move to the 4th X in the bottom row, then to the U at the end of the 4th row, then the L at the beginning of the 3rd row, etc. The result is shown in Figure 5.

L	L	L	L	L
L	L	L	L	L
L	L	U	L	L
U	U	L	U	U
X	X	X	X	X

Figure 4: LUX of a Magic Square of Order 10.

2.2 Print a magic square (Mandatory)

Kana asks you to help them write a MATLAB script to print a magic square, the script is like:

i) Open a file `input.txt` and read an integer n from it. It is grarunteed that the input file only contains an integer $n > 0$.

ii) Open a file `output.txt` with write permission.

iii) Call the function you wrote in the previous section with the integer n read from `input.txt` and print the returned matrix in `output.txt`. If the integer doesn't meet your function (i.e., $n = 2$, or n is even but you wrote the function `mymagicodd`), print some information about it.

68	65	96	93	4	1	32	29	60	57
66	67	94	95	2	3	30	31	58	59
92	89	20	17	28	25	56	53	64	61
90	91	18	19	26	27	54	55	62	63
16	13	24	21	49	52	80	77	88	85
14	15	22	23	50	51	78	79	86	87
37	40	45	48	76	73	81	84	9	12
38	39	46	47	74	75	82	83	10	11
41	44	69	72	97	100	5	8	33	36
43	42	71	70	99	98	7	6	35	34

Figure 5: a Magic Square of Order 10.

The result should be similar to the output of `magic`. You don't need to make the numbers center-aligned (may try it after completing all of other tasks if you wish to.), keep them right-aligned is also fine. You are supposed to use the c-style formatted print function `fprintf`.

Don't forget to close the opened file before your script ends.

Note

Please use monospaced font in MATLAB (by default), or you will find it hard to align the numbers!

It is also recommended to use monospaced font in all of your coding environment. For more information about monospaced font, see Simon's knowledge about Monospaced font.

2.3 Perfect Magic Square Generation (Optional)

After receiving your code on constructing three different types of magic squares, Kana and Haruka are able to read your code and write a function `mymagic` to generate a magic square of arbitrary order. They will be happy if you can write that for them, by forming a group of three from three different sections and combining your code.

It's an optional work since they may finish it after several days without your help. Are you willing to save a week for them?

2.4 Comparing the Two Outputs Using the Terminal (Optional)

As we have mentioned in the previous lab, the terminal is a very useful tool. This time, we will use the terminal to test whether the two outputs we got are the same.

First, after running the **cmd** on Windows or the **terminal** on iOS and Linux, we need to use the **cd** command to get down to the working directory where we keep the MatLab programs as well as the output files. Note that you can see the working directory right above the editor and you can click on it to copy the path to the clipboard.

Then, you can use the **diff** command to compare the two files like `diff file1 file2`. If there is no output, the two files are exactly the same. Moreover, by adding `-w` to the command, we can get rid of

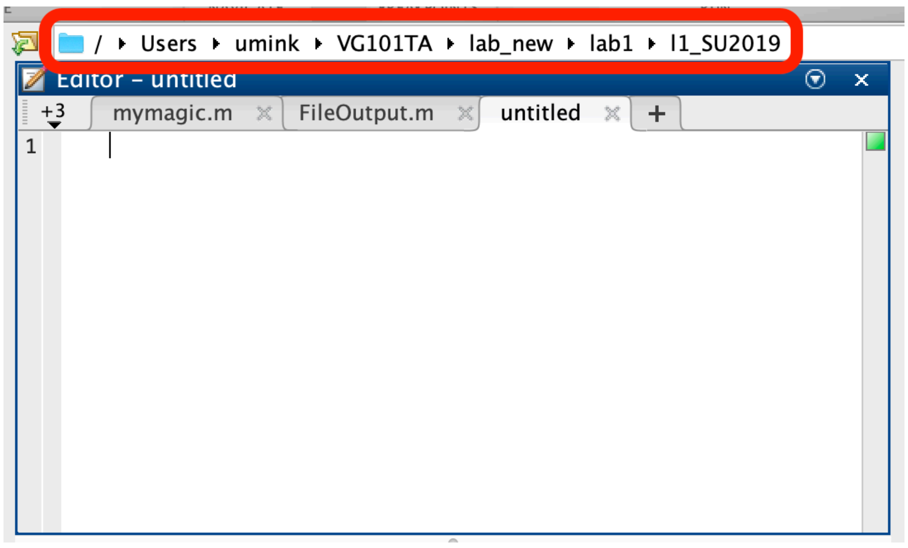
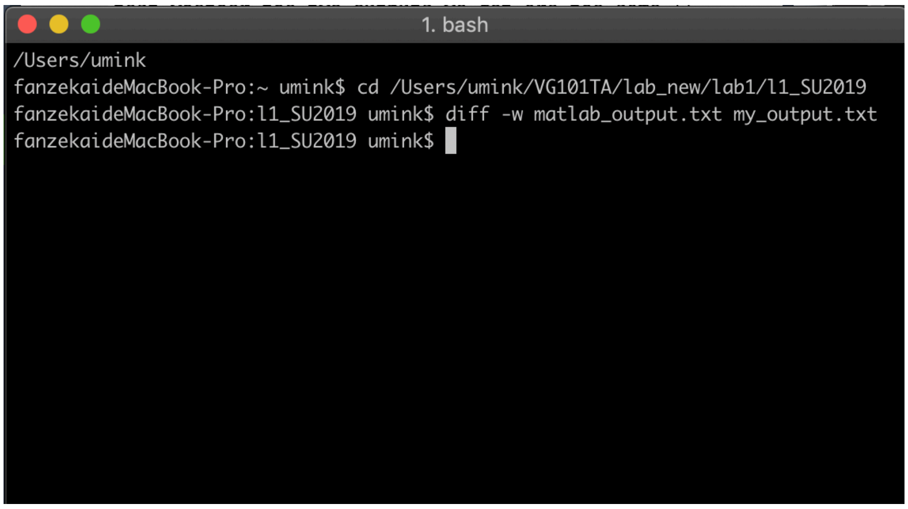


Figure 6: where to find the working directory

the influence of the spaces. After entering 3 in `input.txt` and run `mymagic` in Matlab, we can use the terminal and get the following result. We can see that there's no output, indicating that the files are the same.



```
/Users/umink
fanzekaideMacBook-Pro:~ umink$ cd /Users/umink/VG101TA/lab_new/lab1/l1_SU2019
fanzekaideMacBook-Pro:l1_SU2019 umink$ diff -w matlab_output.txt my_output.txt
fanzekaideMacBook-Pro:l1_SU2019 umink$ 
```

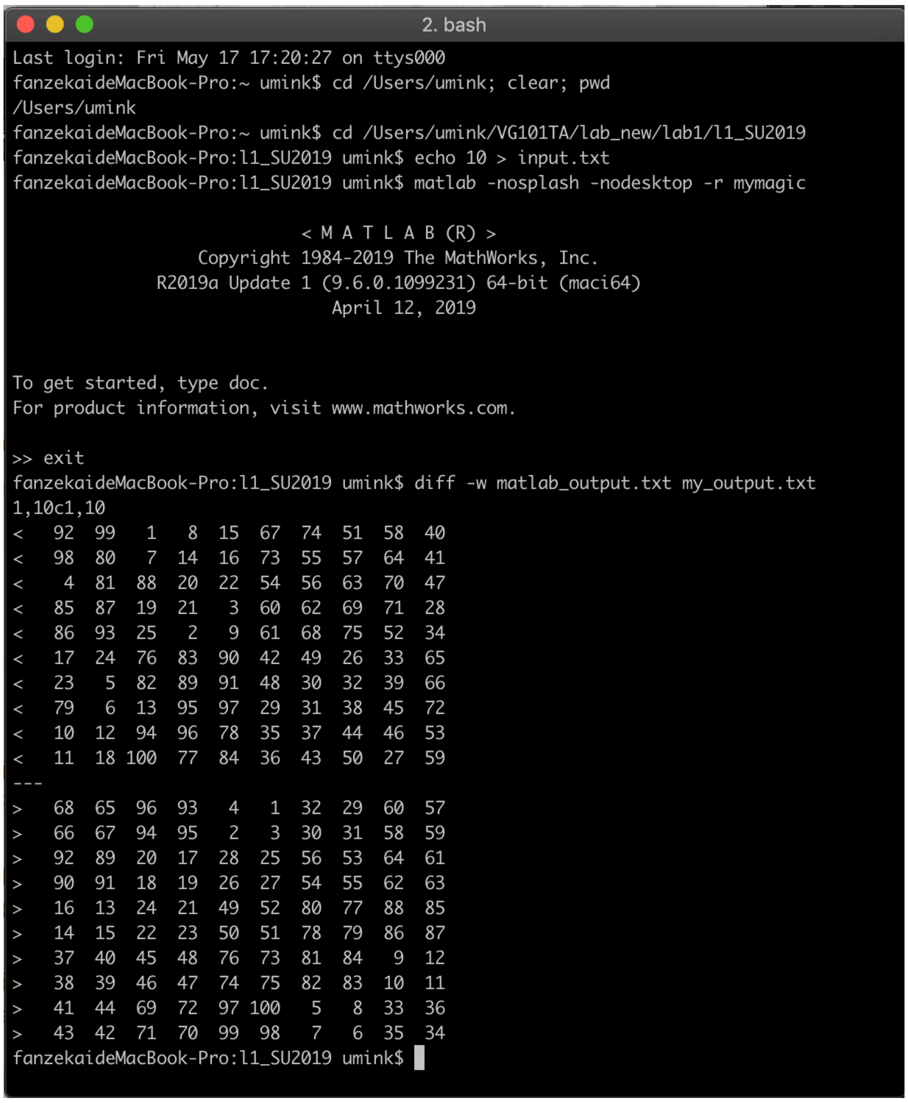
Figure 7: The terminal result

Actually, the terminal can do much more than this. We can do everything from the very beginning. Note that the `matlab` command in the following figure is not always available. While for Linux users, this function can be used upon the installation of MatLab, for iOS and Windows users, you need to configure the environment variables in advance.

We can see that after writting 10 into the input file, the two output files are different. The difference is due to the fact that the function `magic` employs a distinct algorithm to generate the magic square. It's fine since both of the squares are correct.

3 End

You completed Kana and Haruka's request successfully, and proved your mastery in MATLAB. It can be predicted that they are going to ask you for help again in the near future, MAYBE in the next week?



```
Last login: Fri May 17 17:20:27 on ttys000
fanzekaideMacBook-Pro:~ umink$ cd /Users/umink; clear; pwd
/Users/umink
fanzekaideMacBook-Pro:~ umink$ cd /Users/umink/VG101TA/lab_new/lab1/l1_SU2019
fanzekaideMacBook-Pro:l1_SU2019 umink$ echo 10 > input.txt
fanzekaideMacBook-Pro:l1_SU2019 umink$ matlab -nosplash -nodesktop -r mymagic

                            < M A T L A B (R) >
                  Copyright 1984-2019 The MathWorks, Inc.
              R2019a Update 1 (9.6.0.1099231) 64-bit (maci64)
                               April 12, 2019


To get started, type doc.
For product information, visit www.mathworks.com.

>> exit
fanzekaideMacBook-Pro:l1_SU2019 umink$ diff -w matlab_output.txt my_output.txt
1,10c1,10
<    92  99   1   8  15  67  74  51  58  40
<    98  80   7  14  16  73  55  57  64  41
<     4  81  88  20  22  54  56  63  70  47
<    85  87  19  21   3  60  62  69  71  28
<    86  93  25   2   9  61  68  75  52  34
<    17  24  76  83  90  42  49  26  33  65
<    23   5  82  89  91  48  30  32  39  66
<    79   6  13  95  97  29  31  38  45  72
<    10  12  94  96  78  35  37  44  46  53
<    11  18 100  77  84  36  43  50  27  59
---
>    68  65  96  93   4   1  32  29  60  57
>    66  67  94  95   2   3  30  31  58  59
>    92  89  20  17  28  25  56  53  64  61
>    90  91  18  19  26  27  54  55  62  63
>    16  13  24  21  49  52  80  77  88  85
>    14  15  22  23  50  51  78  79  86  87
>    37  40  45  48  76  73  81  84   9  12
>    38  39  46  47  74  75  82  83  10  11
>    41  44  69  72  97 100   5   8  33  36
>    43  42  71  70  99  98   7   6  35  34
fanzekaideMacBook-Pro:l1_SU2019 umink$ 
```

Figure 8: The terminal result