Midterm 1 Review: Example Exams

Question types

1. Loops
    - The most important and possible question in exam;
    - Two types of loop
    - loop within another loop
        1. What should be done in each loop?
        2. Where should you initialize/update/output each variable?
    - Example: 2017 Q1, 2018 Q1, 2019 Q1;
    - In fact, nearly all the questions you will meet in the mid will include loop
2. String/Array/Matrix manipulation
    - Use single quotation for string
    - Array of characters
        - `['abc', 'def'] == 'abcdef'`
    - Connecting strings
    - Travesal the string
    - ASCII code & some tricks
        - `if char >= '0' && char <= '9'`
    - Example: 2017 Q3, 2019 Q4
3. Plot
    - Example: 2017 Q4
4. File IO
    - `fopen` and `fclose`
    - `fscanf` , `fgetl` , and `fprintf` ;
    - Example: 2018 Q4, 2019 Q4

2017 Mid1 Q3: num2English

> 3.(20 points) Write a Matlab function with prototype “function str = num2English(n)” to convert an integer n into the corresponding English words. For example, num2English(143) will return ‘one hundred and forty-three’. Here we assume 0 <= n <= 999999.

This is a question about string manipulation, in fact. The general idea is to convert the input number to a string, and then work on it digit by digit.

```
str = '';
numstr = num2str(n);
```

Here is some questions you may think about when you deal with this question.

1. Is there any pattern when you reading a number (0-999999) in English?
2. In what sequence should you generate the string?
3. How to deal with single digit in different position?
4. How to deal with 11-19?
5. How about zero in each digit?
6. Where should you add dash-line and 'and'?

Possible Answers

1. In fact, we read number in group of three. e.g.
   - *123123 -> 123|123 ->* ***one hundred and twenty-three | thousand and | one hundred and twenty-three***
   - When connecting the groups, we add 'thousand'
   - The other part all the same.
2. Backward, we start from the least significant digit of a number, and concatenat the new result in front of it.

   ```
   length = size(numstr, 2);
   for i=1:length
       digit = numstr(length - i + 1); % read from right to left
       % do something
   end
   ```

3. As we have mentioned, we will process the number in the group of three, so we only need consider three situations:

   ```
   if mod(i, 3) == 1 % 1st or 4th digit
       % do something
   elseif mod(i, 3) == 2 % 2nd and 5th digit
       % do something
   else % remember to add 'hundred' here
       % do something
   end
   ```

   Use `switch` to switch between situations.
4. deal with it when reading the least significant digit (`mod(i, 3) == 1`) in the group of three:

   ```
   if i + 1 <= length && numstr(length - i) == '1' % consider 11-19
       % do something
   end
   ```

5. Do nothing, but do remember to remove meaningles 'and', 'hundred', and 'thousand'.
6. Add dash line when dealing with the second significant digit in a group

```
if numstr(length -  i + 2) ~= '0' && digit > '1'
% if something in previous digit and current is larger than 1
    str = strcat('-', str);
end
```

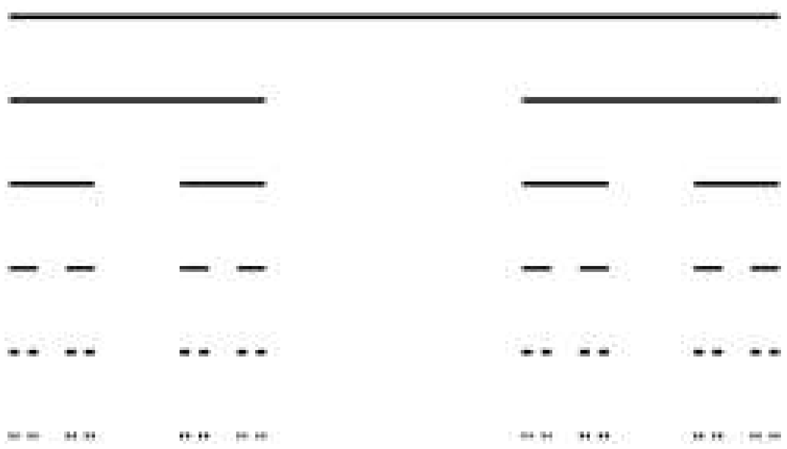
Add 'and' after 'hundred' and 'thousand' if **there is something behind it**.

```
% only present the case for hundred
if numstr(length - i + 2) ~= '0' || numstr(length - i + 3) ~= '0'
% if something before
    str = [' and ', str];
end
```

One thing you may notice is that `strcat[' ', 'abc']` *will not add leading space to the resulting string, use another method here.*

2018 Mid2 Q4: DrawCantor

> 4.(20 points) The following is the description of the famous Cantor set excerpted from https://en.wikipedia.org/wiki/Cantor_set: “The Cantor ternary set is created by repeatedly deleting the open middle third of a set of line segments. One starts by deleting the open middle third (⅓, ⅔) from the interval [0, 1], leaving two line segments: [0, ⅓] ∪ [⅔, 1]. Next, the open middle third of each of these remaining segments is deleted, leaving four line segments: [0, ⅑] ∪ [²⁄₉, ⅓] ∪ [⅔, ⁷⁄₉] ∪ [⁸⁄₉, 1]. This process is continued ad infinitum, where the nth set is
>
> $$C_n = \frac{C_{n-1}}{3} \cup (\frac{2}{3} + \frac{C_{n-1}}{3}) \text{ and } C_0 = [0, 1]$$
>
> The Cantor ternary set contains all points in the interval [0, 1] that are not deleted at any step in this infinite process.”
> The Cantor sets C_0 to C_n can be described graphically using the following figure showing the remaining line segments:



> Write a Matlab function with prototype "function DrawCantor(n)" to draw the above image in a Matlab figure to represent the Cantor sets C0 to Cn, given the function input n >= 0. For example, the above figure shows the result of DrawCantor(5).

This is a plot question, but also a question about loop. With given recurrent relation you should be able to find the new sets of line segments

Here are some questions you may consider when work on this question.

1. How to plot a line in MATLAB?
2. How to store all these line segments (remember that you need a tuple of (x, y) coordinate to draw the line)?
3. How to get the new set of line segments?
4. How to draw multiple lines in one figure?

Possible Answers

1. One possible solution is to use following code:

   ```
   plot([x1, x2], [y1, y2]);
   ```

   Then you will get a line connecting point `(x1, y1)` and `(x2, y2)`.
2. We first consider the x coordinate. One possible solution is to store the `x1` and `x2` seperately, i.e. we have two array, one to store the starting point of all the line segments, the other for the ending point.

   ```
   left = 0; % store the starting coordinate for all segment
   right = 1; % store the ending coordinate for all segment
   ```

   This also give the initial condition.
   And for C_1, we will have:

   ```
   % not real code, just for demo
   left = [0, 2/3]
   right = [1/3, 1]
   ```

And so on.

Recall that a single number in MATLAB is actually 1 by 1 matrix

For y coordinate, we can simply let it decrease from n to 1.

3. To update the line segments, there are in fact two steps according to the recurrent relation:

$$C_n = \frac{C_{n-1}}{3} \cup (\frac{2}{3} + \frac{C_{n-1}}{3}) \text{ and } C_0 = [0, 1]$$

   We first generate the first half $\frac{C_{n-1}}{3}$:

```
newleft = zeros([1, 2 * size(left, 2)]); % size doubled
newright = zeros([1, 2 * size(right, 2)]);
for j = 1 : size(left, 2) % first half of new segments
    newleft(j) = left(j) / 3;
    newright(j) = right(j) / 3;
end
```

   Then we generate the other part:

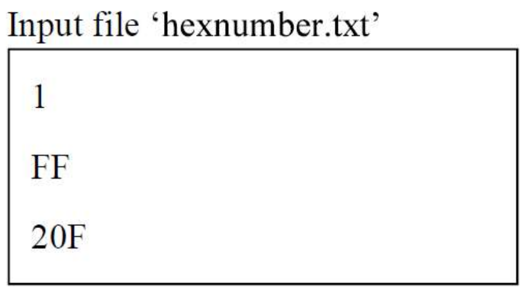
```
for j = 1 : size(left, 2) % second half of new segments
    newleft(j + size(left, 2)) = newleft(j) + (2 / 3);
    newright(j + size(left, 2)) = newright(j) + (2 / 3);
end
left = newleft; % update result
right = newright;
```

   Then you only need to plot the result.

4. Use `hold on;`

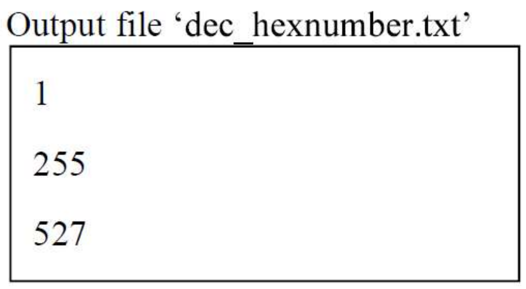
2018 Mid1 Q4: FileHex2Dec

> 4.(20 points) Write a Matlab function with prototype
>
> ```
> function FileHex2Dec(filename)
> ```
>
> to convert hexadecimal numbers in a text file with file name of filename into decimal numbers and save them to another text file. The input text file contains a list of hexadecimal numbers where each line contains only one number. Similarly the output text file contains a list of the corresponding decimal numbers where each line also contains only one number. The following shows the contents of an example input file and the corresponding output file after calling the function:



Input file ‘hexnumber.txt’

```
1
FF
20F
```



Output file ‘dec_hexnumber.txt’

```
1
255
527
```

> Furthermore, the output file name filename filename] should be replaced with the input file name. For example if the input filename is 'hexnumber.txt', then the output file name should be 'dec_hexnumber.txt'.

This is another loop question about fileIO. The problem is how to read and process to result line by line.

Also, there are some question for you to think about:

1. How to know whether you are at the end of the file? If we need a loop, what type should it be?
2. How to read/write your result to a file?
3. How to implement base conversion if you cannot use `hex2dec` ? (left as excise)

Possible Answers

1. Use `feof` function

   ```
   while ~feof(fid) % while not reach the end of the file
       % do something
   end
   ```

2. For reading, you can also use `fgetl` function to get a whole line from the file. For writing, use `fprintf` with one more parameter.

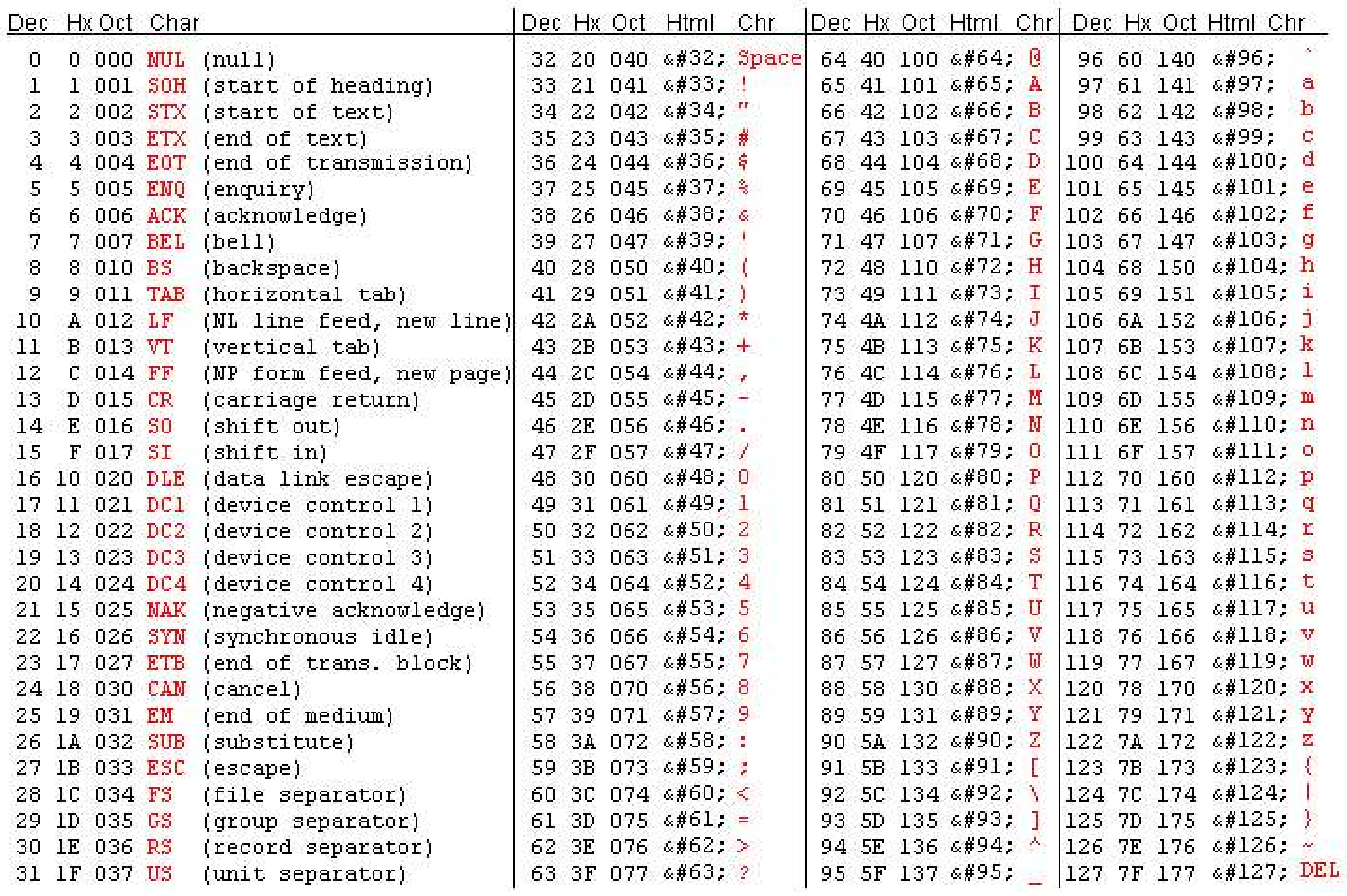
   ```
   line = fgetl(fid); % get a new line from file
   % do somethin to get result in dec
   fprintf(targetFid, '%d\n', dec); % output to target file
   ```

Reminder

1. **For all the functions appear in today's rc. go check the MATLAB official documentation if you don't understand what is it means!**
2. Since this is a open-internet exam, get familiar with all the resources we mentioned.
3. **DO NOT USE ANY CHAT SOFTWARE (EXCEPT ZOOM FOR ONLINE EXAM) DURING THE EXAM!**

Other resources

ASCII code table



Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040	&#32;	Space	64	40	100	&#64;	@	96	60	140	&#96;	`
1	1	001	SOH (start of heading)	33	21	041	&#33;	!	65	41	101	&#65;	A	97	61	141	&#97;	a
2	2	002	STX (start of text)	34	22	042	&#34;	"	66	42	102	&#66;	B	98	62	142	&#98;	b
3	3	003	ETX (end of text)	35	23	043	&#35;	#	67	43	103	&#67;	C	99	63	143	&#99;	c
4	4	004	EOT (end of transmission)	36	24	044	&#36;	$	68	44	104	&#68;	D	100	64	144	&#100;	d
5	5	005	ENQ (enquiry)	37	25	045	&#37;	%	69	45	105	&#69;	E	101	65	145	&#101;	e
6	6	006	ACK (acknowledge)	38	26	046	&#38;	&	70	46	106	&#70;	F	102	66	146	&#102;	f
7	7	007	BEL (bell)	39	27	047	&#39;	'	71	47	107	&#71;	G	103	67	147	&#103;	g
8	8	010	BS (backspace)	40	28	050	&#40;	(	72	48	110	&#72;	H	104	68	150	&#104;	h
9	9	011	TAB (horizontal tab)	41	29	051	&#41;	)	73	49	111	&#73;	I	105	69	151	&#105;	i
10	A	012	LF (NL line feed, new line)	42	2A	052	&#42;	*	74	4A	112	&#74;	J	106	6A	152	&#106;	j
11	B	013	VT (vertical tab)	43	2B	053	&#43;	+	75	4B	113	&#75;	K	107	6B	153	&#107;	k
12	C	014	FF (NP form feed, new page)	44	2C	054	&#44;	,	76	4C	114	&#76;	L	108	6C	154	&#108;	l
13	D	015	CR (carriage return)	45	2D	055	&#45;	-	77	4D	115	&#77;	M	109	6D	155	&#109;	m
14	E	016	SO (shift out)	46	2E	056	&#46;	.	78	4E	116	&#78;	N	110	6E	156	&#110;	n
15	F	017	SI (shift in)	47	2F	057	&#47;	/	79	4F	117	&#79;	O	111	6F	157	&#111;	o
16	10	020	DLE (data link escape)	48	30	060	&#48;	0	80	50	120	&#80;	P	112	70	160	&#112;	p
17	11	021	DC1 (device control 1)	49	31	061	&#49;	1	81	51	121	&#81;	Q	113	71	161	&#113;	q
18	12	022	DC2 (device control 2)	50	32	062	&#50;	2	82	52	122	&#82;	R	114	72	162	&#114;	r
19	13	023	DC3 (device control 3)	51	33	063	&#51;	3	83	53	123	&#83;	S	115	73	163	&#115;	s
20	14	024	DC4 (device control 4)	52	34	064	&#52;	4	84	54	124	&#84;	T	116	74	164	&#116;	t
21	15	025	NAK (negative acknowledge)	53	35	065	&#53;	5	85	55	125	&#85;	U	117	75	165	&#117;	u
22	16	026	SYN (synchronous idle)	54	36	066	&#54;	6	86	56	126	&#86;	V	118	76	166	&#118;	v
23	17	027	ETB (end of trans. block)	55	37	067	&#55;	7	87	57	127	&#87;	W	119	77	167	&#119;	w
24	18	030	CAN (cancel)	56	38	070	&#56;	8	88	58	130	&#88;	X	120	78	170	&#120;	x
25	19	031	EM (end of medium)	57	39	071	&#57;	9	89	59	131	&#89;	Y	121	79	171	&#121;	y
26	1A	032	SUB (substitute)	58	3A	072	&#58;	:	90	5A	132	&#90;	Z	122	7A	172	&#122;	z
27	1B	033	ESC (escape)	59	3B	073	&#59;	;	91	5B	133	&#91;	[	123	7B	173	&#123;	{
28	1C	034	FS (file separator)	60	3C	074	&#60;	<	92	5C	134	&#92;	\	124	7C	174	&#124;	|
29	1D	035	GS (group separator)	61	3D	075	&#61;	=	93	5D	135	&#93;	]	125	7D	175	&#125;	}
30	1E	036	RS (record separator)	62	3E	076	&#62;	>	94	5E	136	&#94;	^	126	7E	176	&#126;	~
31	1F	037	US (unit separator)	63	3F	077	&#63;	?	95	5F	137	&#95;	_	127	7F	177	&#127;	DEL

`help` function in MATLAB

GOOD LUCK~