EGR 141: Final Project

Summary: The goal of this project is to take everything you've learned this year and apply it to some interesting problems.

- Each of the following problems should have a script and, possibly, some functions associated with them.
- For each problem, the script file should be called something appropriate, such as *final_1_yourName.m*
- Include any functions that you needed to create in order to complete the problem. Name them appropriately. I would suggest using local functions within your script.
- Make the first line of each script: *clear; close; clc;*
- Do *not* output any large-sized vectors to the screen.
- Make sure to test your functions on *all* examples in the given problem.
- *Each problem* should be in its own script.
- Note that example output for each problem is not necessarily correct output.
- If my example output "lines up nicely" then your output should as well.
- Choose four of the following problems. You may choose only four. If you hand in more than four, you will receive a zero on the final project.
- Absolutely no projects will be accepted late (I cannot grade them in time if you hand them in late).

Optimization

Create a optimization problem solver function that works for the specific case of

Goal: Find min/max of equation of the form $f(x, y)$, $x \geq 0$ *and* $y \geq 0$

Given: Some relationship between x *and* y, *given by* $g(x, y) = 0$ *and some bounds on* x, $a \leq x \leq b$.

The optimization problem should be solved using the standard process learned in Calculus. The input into your function should be anonymous function f and g along with bounds a and b (note that a and b could be infinite). The output should be two variables, each of length 2. The first corresponds to the (x, y) values at the absolute/global min. The second corresponds to the (x, y) values at the absolute/global max. If there are multiple global minima/maxima, you need only return one (x, y) pair. Test your algorithm of the following problems, printing the results to the screen, indicating locations of min/max values as well as the actual minimum and maximums.

1. A rectangle is to be inscribed in the ellipse $\frac{x^2}{9} + y^2 = 1$. Find the dimensions of the largest such rectangle as well as the maximum area.
2. A rectangular box with a square base, open top, and volume of $1728\,\text{in}^3$ is to be constructed. Find dimensions of the box that minimize surface area. What is the minimum surface area?

```
Final Project - Optimization Solver
Problem statement:
 Given one side of a garden is against your house,
 what is the largest garden you can construct with
 100 ft of fencing?
Min/Max of f(x,y)=x*y  subject to 2*x+y-100=0
 where 0<=x<=50
The min area is 0 at (0,100)
The max area is 1250 at (25,50)
```

Mad Libs

The file *wordsWithTypes.txt*, is a tab-delimited text file containing two columns of strings. The first column contains a word type (*noun*, *verb*, *adjective*, and *adverb*), while the second contains the corresponding word. For the below paragraph, fill in each (number) with the given word type, with the given word properties.

Getting started with (1) and making a game can prove to be a (2) task, so as an independent designer, here are some tips that can help you get a head start in the wide (3) of video game development. The (4) thing that can help you on your gaming journey is knowing the (5) simple role groups. Designers are responsible for the (6) and all the (7) stuff the player can do in their game, such as (8) a room that needs you to find multiple (9) . Artists have it rough, working day in and day out (10) and (11) art assets and (12), and sometimes also work on creating (13)! Producers are the task managers, responsible for keeping a team (14), (15) and overseeing all other (16) of the game. Then there's the programmers, who just sit at (17) and type fancy syntax (18) into a (19).(Source: Wordblanks.com)

1. ING Ending Verb	6. THS Ending Noun	11. ING Ending Verb	16. Noun
2. Adjective	7. Adjective	12. Noun Starting S	17. VES Ending Noun
3. Noun	8. ING Ending Verb	13. Noun	18. Noun
4. Adjective	9. SSES Ending Noun	14. Adjective	19. Noun
5. Number	10: ING Ending Verb	15. Adjective	

Draw randomly from all possibilties (with the correct properties). For the number, draw randomly from 0 to 99.

```
Final Project - MadLibs
Getting started with FIFING and making a game can prove to be a TAI
task, so as an independent designer, here are some tips that can
help you get a head start in the wide FELLIES of video game development.
The SYLPHY thing that can help you on your gaming journey is knowing the
18 simple role groups. Designers are responsible for the WORDSMITHS and
all the DAPPLE stuff the player can do in their game, such as REEVING a
room that needs you to find multiple KISSES . Artists have it rough,
working day in and day out HINGING and BABBLING art assets and SORENESS, and
sometimes also work on creating MOONSHOT! Producers are the task managers,
responsible for keeping a team NITID, TESTY and overseeing all other
BARGEBOARDS of the game. Then there's the programmers, who just sit at SALVES
and type fancy syntax MUST into a UNCTION.(Source: Wordblanks.com)
```

Roman Numeral Conversion

Create a function *numOut = romanNumeralConverter(numIn)* which converts to/from Roman numerals and decimal integers. A few notes and requirements:

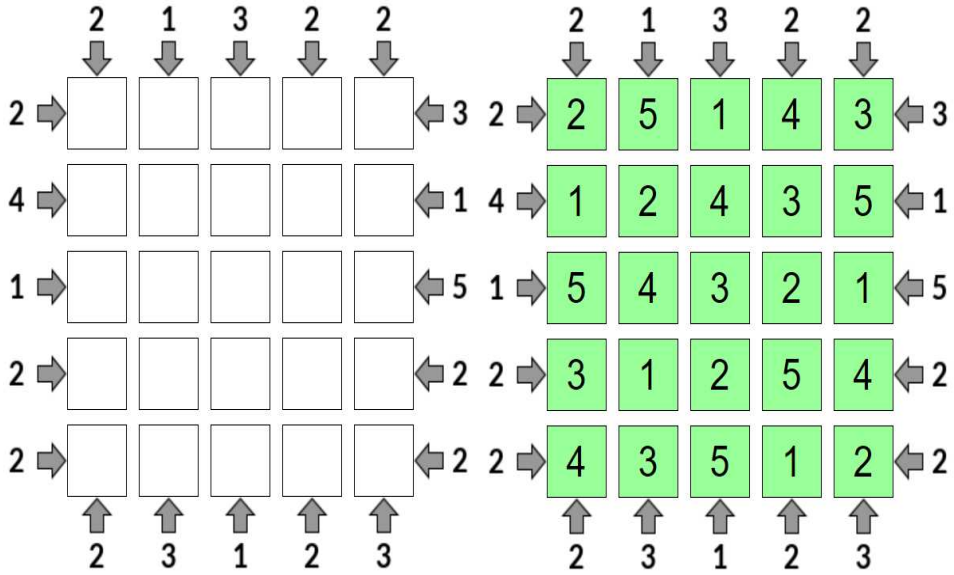
- Not this is a *single* function that deals with either conversion
- Decimal input/output should be numerical (for example *1234 or 83)*
- Roman numeral input/output should be a string (not a character vector) (for example *"MLXXIX"* or *"MCLV"*)
- Only integers between 1 and 3999 (I to $MMMCMXCIX$) are allowed as input
- No if statements, switch statements, or any other decisions statements are allowed to be used as part of the conversion. They may *only* be used for input verification (to see if the input is in a valid range, or to see if it is a character vector, integer, etc)
- In your script, test on each of the following: *1900, 649, 2568, MDCCC, MMDCCXXXVIII, CCLV*
- You will be graded on efficiency.

```
Final Project - Roman Numeral Converter
2630 is converted to MMDCXXX
MMMCLXX is converted to 3170
```

Skyscraper Puzzle Solver

Create a routine that solves any 4×4 or 5×5 Skyscraper puzzle. Recall that the goal is to determine the heights of buildings in the given grid. The numbers indicate how many buildings are visible from each direction. Note that taller buildings will block any shorter buildings behind them. Each row and column must have exactly one building of each height.

Below is an example starting problem, along with the solution. Note that the solution is unique.



Assume you are given the top, bottom, left, and right numbers (all of them). You may choose to format these however you want. Output the filled-in solution to the screen. A few notes/requirements:

- Your code should work for both a 4×4 and a 5×5, and should be programmed universally. In other words, you should not have a "4×4 code block" and a "5×5 code block".
- Your code should not run for more than 30 seconds (wall time, eg, tic-toc time)
- The *combvec* function could be useful.
- In many fonts, the arrow characters are around 8592 in the character set (in many fonts, they don't exist).
- Note that my example is a tiny bit "off" in terms of lining up (due to characters not taking up an entire space). If your table is less than one character width askew in one or two places, that is perfectly fine.

Inside your script, test on both the 4×4 and 5×5 daily puzzles found here (https://www.brainbashers.com/skyscrapers.asp) for one of the days in May. In the comments, indicate which day you chose.

```
Final Project - Skyscraper Solver
   | 2 | 5 | 2 | 3 | 1 |
   | ↓ | ↓ | ↓ | ↓ | ↓ |
2→| 4 | 1 | 2 | 3 | 5 |←1
2→| 3 | 2 | 5 | 1 | 4 |←2
1→| 5 | 3 | 1 | 4 | 2 |←3
3→| 2 | 4 | 3 | 5 | 1 |←2
2→| 1 | 5 | 4 | 2 | 3 |←3
   | ↑ | ↑ | ↑ | ↑ | ↑ |
   | 3 | 1 | 2 | 2 | 3 |
```

Root Finding Methods

Create three functions that apply the bisection method (recall lab 6), Secant method, and Newton's method (lab 8). Input should be consistent with previous labs for both the bisection and Newton's method. Secant method should have input of a function handle, two guesses (x_0 and x_1) and a tolerance. You may look up information on Secant method but, of course, do not look up code.

Solve each of the following problems using each of the three methods.

1. A root of $f(x) = x^2 - 3$
2. A local max of $g(x) = e^{\cos x} - \sin x + x^2$
3. An inflection point of $h(x) = \ln\left(x^2 + 2\right)$

Use an interval $[a, b] = [0, 2]$. For the Secant method use $x_0 = a$ and $x_1 = b$. For Newton's method, use $x_0 = b$. For all, use a tolerance of 10^{-7}.

For each problem and each method, print out the iteration (i), current approximation for that iteration (x_i), the absolute value of the error between the current approxmation and the final approximation ($e_i = |x_i - x_N|$) and the ratio between the next approximation and the current one (e_{i+1}/e_i). In your comments, include an analysis of the error ratios for each method (do they stay approximately the same, decrease slowly, etc).

```
Final Project - Root Finding Methods
Roots of f(x)=cos(pi*x), tol = 1e-04
  Bisection Method over [5/12,2/3]
Itr       Value          Err     ErrRatio
  1   0.5416667   0.0416870     0.49927
  2   0.4791667   0.0208130     0.50147
  3   0.5104167   0.0104370     0.49708
  4   0.4947917   0.0051880     0.50588
  5   0.5026042   0.0026245     0.48837
  6   0.4986979   0.0012817     0.52381
  7   0.5006510   0.0006714     0.45455
  8   0.4996745   0.0003052     0.60000
  9   0.5001628   0.0001831     0.33333
 10   0.4999186   0.0000610     1.00000
 11   0.5000407   0.0000610     0.00000
 12   0.4999797   0.0000000     -
  Secant Method with guesses 5/12 & 2/3
Itr       Value          Err     ErrRatio
  1   0.5019370   0.0019370     0.04777
  2   0.4999075   0.0000925     0.00000
  3   0.5000000   0.0000000     -
  Newton's Method with initial guess 2/3
Itr       Value          Err     ErrRatio
  1   0.4828904   0.0171261     0.00000
  2   0.5000165   0.0000000     -
```

Slot Machine

Using the supplied code, create a working simulation of a slot machine. The code given does the following:

- When running the command *slot*, displays the structure of a slot machine, without any images displayed on the rollers
- Runs the code found in *playGameOfSlots* when the Spin button it hit.

You must complete the rest of the slot machine "game" implementing code that:

- Controls the change of the rollers, as defined in the given *playGameOfSlots.m* file
- Defines a "payout" amount depending on the end result of the slots and the current bet amount.

You may not change any of the code in the main *playGameOfSlots* code or the constucted *slots.mlapp* GUI application. You may only change the functions indicated.

This will require you to learn quite a bit about GUIs in MATLAB. I would suggest going online to get some inspiration for how a slot machine game works.