MA 302 – Spring 2019
Final project (due Saturday, May 4)

Instructions:

1. The code submitted should be your own creation.
2. You may ONLY consult MATLAB's documentation or the notes from class.
3. You may not speak to any other students about any aspects of your code.

In short, do not plagiarize. It is unfair to your classmates and a violation of Loyola's honor code and will result in failure of the project and the course.

Problem 1a: Estimating π - How about throwing a bunch of random points at it?

Write a MATLAB function with heading

```
function approx_val_pi = approx_pi(N)
```

that estimates π by generating `N` random points in the square $[-1,1]\times[-1,1]$, which has an area of 4. The area inside the circle of radius 1, which is enclosed by the square is π. Therefore the fraction of points that land inside the circle gives an estimate of $\frac{\pi}{4}$.
Testing

Write a script to perform the following tests and include your observations in your project summary file:

1. Compute the absolute error, `abs(approx_val - pi)` for $N = 10^k$, $k = 1, \ldots, 7$.
2. Plot on the same figure:
   (a) The $\log_{10}$ of the error as a function of the $\log_{10} N$
   (b) The least squares fit line for the error and values of N on the log scale.

   Save your image as `proj_p1.jpg` and include this with your submission.
3. The error in approximating π using this technique can be written as

$$E = CN^p$$

   where C is a constant independent of N. Use the equation of your linear fit model to determine p.

Problem 1b: Estimating π - Arcus Tangens, an elegant approach

John Machin (1680-1752) discovered the following identity

$$\pi = 16\arctan\left(\frac{1}{5}\right) - 4\arctan\left(\frac{1}{239}\right)$$

For centuries, Machin's formula served as a primary method for finding digits of π. In fact, as late as 1973, one million digits of π were evaluated by Guilloud and Bouye (Arndt, J and C. Haenel (2001). *Pi-Unleashed*. Springer) using a version of Machin's formula. Write a function to approximate π using the following approximation.

$$\pi \approx T_n = 16P_n\left(\frac{1}{5}\right) - 4P_n\left(\frac{1}{239}\right)$$

where,

$$P_n(x) = \sum_{k=0}^{n} \frac{(-1)^k}{2k+1} x^{2k+1}$$

is the *nth*-degree Taylor polynomial for $\arctan x$ centered at $x = 0$.

Testing
Write a script to test your function as follows:

1. Compute the absolute error, $|T_n - \pi|$ for $n = 1, 3, 5, 7, 9, 11$ and include a table of errors and values of n in your summary file.
2. Provide a short comment on the two approaches for approximating π, which one do you prefer? Why?

Problem 2: Population dynamics – Euler's take

Suppose $x(t)$ is the population of a species and consider the population model

$$\frac{dx}{dt} = rx\left(1 - \frac{x}{k}\right) - \frac{x^2}{1+x^2}$$

The first term on the right-hand side is known as the logistic growth term. This term comes from the assumption that for small population levels, the population will grow at a rate proportional to the current level, while, for large populations, limited resources will cause the growth rate to decrease and eventually become negative. The parameters r and k are called the natural growth rate if the population and the environmental carrying capacity, respectively. The second term on the right-hand side represents harvesting/predation of the species by some other species (e.g. fish being caught by fisherman).

1. For $r = 4$ and $k = 20$, write a script that uses Euler's method to determine the eventual population level reached from an initial population of 2.44 using a time step of $h = 0.1$ weeks.

   Implementation note
   During the semester we implemented Euler's method to run on a specific time in a specific interval $[a, b]$. In this case you will need to run Euler's method until the population stops changing, one way to achieve this is to run Euler's method until the difference between the solution at successive time steps is less than `1.0e-06`.

2. Repeat part 1, but with the initial population changed to 2.40.
3. Provide a plots (`proj_p2a.jph, proj_2b.jpg`) of the population in parts (a) and (b) and comment on your results in your summary file.

Problem 3: A game, "fun"!

Write a MATLAB script in which the user will play the following dice game against the machine. Your script should take input from the user using the `input` command and output appropriate messages to update the user regarding the actions of the machine as the game proceeds.
Rules of the dice game
Object: get to 100 points before your opponent.
A player's turn involves rolling a standard six-sided die.

1. If anything other than a 1 is rolled, the number rolled is added to the player's subtotal. The player can choose to roll again, or stop their turn and the subtotal is added to the player's total.

2. If a 1 is rolled, the player's turn is over and no points are added to the player's total.

For the user's turn, your script will roll the die and display the number rolled. If the number is 2 through 6, your script should display the current subtotal and ask the user if they want to roll again.
For the computer's turn, if the number 2 through 6 is rolled, the computer will decide whether to roll again. This could be an initial fixed probability p of rolling at each turn that will have different probabilities depending on the current total score and subscores. Feel free to creative with your algorithms but you don't need to come up with a fancy optimal algorithm. The requirement is that you are able to describe and justify your choice of algorithm.
At the start of each turn (user or computer), the command window is cleared and both player's totals are displayed. For each roll of the dice, the roll, current subtotal and the current total for both players is displayed. The first player to reach 100 points wins the game.

Submission
Your description should include the following:

1. A short description of how a player interacts with your game.

2. A description of how the machine decides whether to roll or not.

3. Text from one complete run of your game.

Problem 4: May the $4th$ be with you!

Download the files `yoda.m` and `yodapose_low.mat`. Read and understand the code in `yoda.m`, run it and then modify the code to perform the following:
Rotate the image by θ radians about the y-axis continuously by $\theta = \dfrac{\pi}{24}$ until the image has made two full rotations about the y-axis. The matrix R below should do the job.

$$R = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix}$$

If you find yourself writing a lot of code for this problem, you have gone too far!

Submission of Project

Place all your files (`m-files, summary.txt, diary.txt`) in a folder named `lastname_project` and zip the folder to create a file `lastname_project.zip`. Email your zip file `lastname_project.zip` to `pchidyagwai@loyola.e`
with subject `MA302_project`.