MA 302 – Spring 2020
Final project (due Friday, May 1)

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 1: 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 $\frac{\pi}{4}$.

Hints

1. Start by drawing a picture of the scenario - you have a square with $-1 \le x \le 1$ and $-1 \le y \le 1$. The area of the square is 4 and we can inscribe a circle of radius 1 inside the square.
2. Now if we generate random pairs of coordinates (x, y) inside the square, some will land inside the circle others will not. Let N be the total number of points generated and N_c be the points inside the circle then
$$\frac{N_c}{N} \approx \frac{\text{area of circle}}{\text{area of square}}$$
so we can use this ratio to estimate π.

Testing

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

1. Assuming `approx_val` is your approximation to *pi*, compute the absolute error, `abs(approx_val ... - pi)` for $N = 10^k$, $k = 1, \ldots, 7$.
2. Plot on the $\log_{10}$ of the error as a function of the $\log_{10} N$. Save your image as `proj_p1.jpg` and include this with your submission.

Problem 2: 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 sub scores. 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 3: May the force be with you - I hope to see you all in the near future

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`.