CS1112 Fall 2022 Project 2 due Monday 9/19 at 11pm

You must work either on your own or with one partner. If you work with a partner you must first register as a group in CMS and then submit your work as a group. *Adhere to the Code of Academic Integrity.* For a group, "you" below refers to "your group." You may discuss background issues and general strategies with others and seek help from the course staff, but the work that you submit must be your own. In particular, you may discuss general ideas with others but you may not work out the detailed solutions with others. **It is not OK for you to see or hear another student's code and it is certainly not OK to copy code from another person or from published/Internet sources**. If you feel that you cannot complete the assignment on you own, seek help from the course staff.

Objectives

Completing this project will solidify your understanding of Chapters 2 and 3 in *Insight* (`for`-loops, `while`-loops, nested loops). You will continue to explore MATLAB graphics.

Ground Rule

Since this is an introductory computing course, and in order to give you practice on specific programming constructs, some MATLAB functions/features are forbidden in assignments. Specifically, the use of arrays and the `break` and `continue` commands is *not* allowed in this project. Only the built-in functions that have been discussed in class and in the assigned reading may be used. For this project, do not define your own functions; we will do that later in Project 3.

1 My calendar

Write a script `monthCal` that prints a one-month calendar. Your script should solicit input for the number of days in the month and the starting day-of-the-week. The output from an example run of the script is shown below (user input is shown in *italics*):

```
Number of days:  31
Starting day-of-the-week (1=Mon, 7=Sun):  2

  Su Mo Tu We Th Fr Sa
         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
```

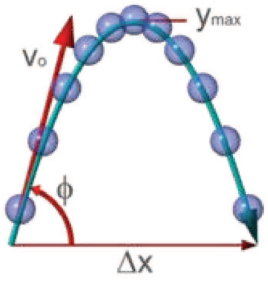
For the user's convenience when inputting the starting day-of-the-week, Monday is day 1, Tuesday is day 2, . . . , Sunday is day 7. However, for printing the calendar you must use Sunday as the first day of the week. The output dates should line up neatly (right-justified) as shown above, but the format doesn't have to be exactly the same.

Hints: (1) Use a `for`-loop to count from 1 to n where n is the number of days. (2) You need to set an appropriate "test" to determine when to begin a new line.

Submit your file `monthCal.m` on CMS (after registering you group).

2 Trajectory of a golf ball

In your physics course you might have studied the motion of a projectile. Now you will write a simulation to plot the trajectory of a golf ball in flight, subject to air resistance (drag). Without air resistance, one expects the trajectory to be a parabola due to Earth's gravity, with equal time for ascending and descending, as shown in the diagram on the right. What is the effect of air resistance? You will find out!



We consider the golf ball to be a unit mass and air resistance to act in the opposite direction of motion. Further we assume that air resistance is proportional to the square of the velocity of the projectile. With these assumptions, the relevant equations are

$$\begin{aligned} v_x &= dx/dt \\ v_y &= dy/dt \\ dv_x/dt &= -k \cdot v_x \sqrt{v_x^2 + v_y^2} \\ dv_y/dt &= -k \cdot v_y \sqrt{v_x^2 + v_y^2} - g \end{aligned}$$

where v_x and v_y are the components of the velocity in the x- and y-directions, k is the coefficient of air drag, and g is the gravitational constant. The golf ball is initially at $x = 0$ and $y = 0$ and is launched with some initial velocity at an angle ϕ measured from the x-axis.

Download the file `golfBall.m` from the *Projects* page. Read and run it. Since the simulation code is missing the output includes only a figure window with axis labels and "dummy values" printed to the Command Window. You will complete the simulation and replace the dummy values with the actual values calculated in the simulation.

Here are the details of our simulation:

- The constants, parameter values, and initial conditions are given in the provided code. Develop your simulation with the given values but you can (and will) experiment with different values later.
- The simulation begins with time $t = 0$ and position $x = 0$ and $y = 0$ and ends when the golf ball lands (y gets back to, or passes, zero) or the maximum allowed simulation time has been reached, whichever happens first.
- Given the initial velocity v and launch angle ϕ, the initial velocities in the x- and y-directions are $v_x = v \cos\phi$ and $v_y = v \sin\phi$
- We use *differencing* to represent the (continuous) derivatives. For example, instead of $\frac{dx}{dt}$ we consider

  $$\frac{x_{\text{new}} - x_{\text{current}}}{\triangle t}$$

  where $\triangle t$ is a discrete time step. With this discrete representation, at each time step, i.e., each step of the simulation, we can compute the new velocities and positions as follows:

  $$\begin{aligned} v_{x\text{new}} &= v_x - \triangle t \cdot k \cdot v_x \sqrt{v_x^2 + v_y^2} \\ v_{y\text{new}} &= v_y - \triangle t \cdot (k \cdot v_y \sqrt{v_x^2 + v_y^2} + g) \\ x_{\text{new}} &= x + v_x \cdot \triangle t \\ y_{\text{new}} &= y + v_y \cdot \triangle t \end{aligned}$$

- Starting at $t = 0$ and after each time step, plot the location of the golf ball. Recall that the command `plot(a,b,'ro')` draws a marker (red circle) at position (a,b). Choose any marker and color you like. Add a pause of 0.01 seconds (`pause(.01)`) after the plot command so that the plotting is animated when the simulation is executed.

Run the code you have so far to make sure that the simulation works! At this point you should have removed all the dummy values except perhaps those for variables `ascendTime` and `descendTime`. Does the trajectory make sense (a curve that opens down, starting and ending at around $y = 0$)? Do the displayed values in the Command Window look correct?

- Add more code to the simulation to compute the durations of ascent and descent. Do this using code based on the simulation here—do not look up equations from physics textbooks! Make sure all the dummy variable values are removed.
- Finally, modify the parameter values `phi`, `v`, `k`, and `maxTime` to see how the trajectory changes. Once you change the parameter values, the complete trajectory may not be shown since the axes are set to 120m wide and 100m tall and `maxTime` may be "too short," but you can use the values displayed in the Command Window to deduce the trajectory. Add a comment at the end of the script to answer these questions:

  1. Which of these launch angle results in the longest horizontal range, $\pi/3, \pi/4, \pi/5, \pi/6$ (without changing the other parameter values)?
  2. How does the flight time change with the launch angle? Answer in one sentence.
  3. What is the shape of the trajectory when $k = 0$?

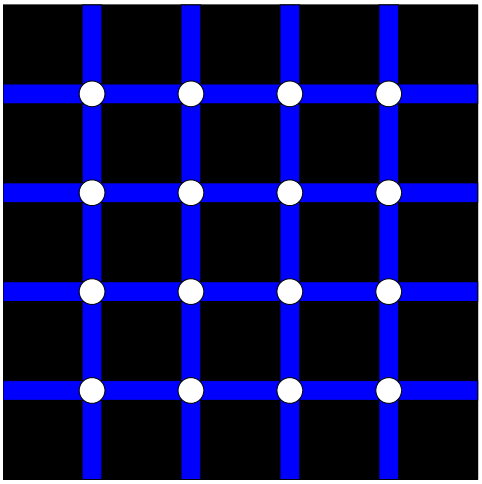
  Answer the above questions simply by *running the simulation* with different parameter values. Do not write more code.

Before submitting your file `golfBall.m` on CMS, change all the parameter values back to the original values given: `k=0.02`, `maxTime=10`, `phi=pi/4`, and `v=100`.

Submit your file `golfBall.m` on CMS (after registering you group).

Problem 3 on the next page involves a diagram that has a flickering effect (an optical illusion). If the flickering could cause any discomfort or ill effect, do not look at the next page and instead ask the instructor for a different problem to solve **no later than Friday, Sept 16**.

3 The "Scintillating Grid"



Are your eyes playing tricks on you? Are some of those white disks in the diagram *flickering*? The diagram on the left is an optical illusion called the "Scintillating Grid." If you focus on a disk at a particular intersection, the neighboring disks appear to flicker. Cool! Now stop staring at it ...

Write a script `illusion` that solicits n, the number of squares along each side and draws the "Scintillating Grid." Assume that n is a positive integer value greater than 1. The example on the left has $n = 5$. Approximate the proportion that you see in the diagram and experiment with different colors to find your favorite scheme (that shows off the illusion).

Parameterize your code, i.e., identify the main properties (values) that define the diagram and name them as variables—parameters. Furthermore, choose *one* property to be the "root" and make the other properties dependent on it. For example, let's say that I see two important properties (there are more in this problem): gap width and square length. Instead of "hard coding" the variable values, for example as `g=1; s=4;`, I should make one variable dependent on the other, e.g., `g=1; scaleS=4; s=scaleS*g;`. Parameterization is an important concept in design and computational engineering. Choosing parameters wisely allows you to easily experiment with different values to "tune" your model and keeps your code easily maintainable.

Use the provided functions `DrawRect` and `DrawDisk`. Download the files from the Projects page and *put them in the same folder as your script* `illusion`.

Graphics note: Use the figure window setup demonstrated in Lecture 7. Recall that `axis equal off` gives equal scaling in the horizontal and vertical axes and hides the axes values. During program development you may wish to see the numeric axes for debugging; in that case use the command `axis equal` to have the axes "on" instead of turning "off" the axes with the command `axis equal off`.

Submit your file `illusion.m` on CMS (after registering you group).