Due: Friday, March 23 at 4 PM

- For this assignment you will create 3 files; `hw6Ober.m`, `chaosOber.m`, and `snowflakeOber.m` where "Ober" is replaced by the first four letters of your last name.
- This SCRIPT FILE `hw6Ober.m` should be formatted in such a way to be "publishable" as a webpage. Your webpage should appear on the G-drive in the same folders as your previous assignments. Within a browser, copy and paste the URL for your page into Moodle by the due date/time.
- All other directions are the same as the previous assignments.

TIP 1: running the fractals will take some time, so **do not publish until you know you have them working**, and when you do publish BE PATIENT!

TIP 2: Remember that the output of commands aren't shown until you have a `%%`, so you may want to have lines of just `%%` between some of the commands so it publishes a set of commands or comments, then creates output, publishes the next set of commands or comments, then output, etc.

To turn in:

1. The URL of the HTML file created by publishing your m-file submitted via Moodle. Have the URL appear in the notes part of turning in the assignment and make it a clickable link.
2. The m-files (both `hw6Ober.m` and all of the function files) uploaded to Moodle by the due date/time.

---

1. **Chaos Game** We will simulate a modified version of the Chaos game (discussed in class) and use complex numbers. We will use use a square instead of a triangle and have eight "vertices" that are the corners and midpoints of the sides. Our *seed* will be a complex number generated by the `rand` command for the real and imaginary values of the seed. You will create a function file called `chaosOber.m` that has one or more inputs. The first input is a positive integer n that will determine the number of "rolls of the die" there will be in the Chaos Game. The additional inputs, which are optional, will be marker specifications for drawing the points. The output of the function will be the time elapsed (generated by the commands `tic` and `toc`) to play the game. Your function file will do the following.
   (i) Capture the start time with `tic`.
   (ii) Perform the error check on the function input n and use the `error` command with appropriate message.
   (iii) Store the vertices of the square in a vector: 1, $1+i$, i, $-1+i$, -1, $-1-i$, $-i$, $1-i$ (you may want to add extra 1 at the end to make it easier to draw...) Draw the square in a solid black line.
   (iv) Store the default colors using the commands
   `colors = lines(8); colors(8,:)=[0,0,0];`
   (v) Draw the vertices using the "o" marker and color them using the default colors stored in `colors`. To make them more visible, specify the `MarkerFaceColor` and `MarkerEdgeColor` to be the appropriate color. You may want the `MarkerSize` to be larger than the default. (Hint: a `for` loop may be useful!)

(vi) Preallocate a vector to the appropriate size to contain the "seed" and the n points generated by the game.

(vii) Generate the seed to be in the first element of this vector by using the `rand` command. This seed should be a complex number. (How would we get a random complex number using the `rand` command?)

(viii) Generate n rolls of an *eight-sided die* in a vector `rolls`.

(ix) Now play the Chaos game for n turns: Using the `rolls` vector to determine the chosen vertex, the next element of your vector of points will be 1/3 of the distance between the previous element and the chosen vertex.

(x) After all of the turns are completed (thus the vector has been filled with the complex numbers obtained from the turns of the Chaos Game), either plot the points in the vector as periods (`'.'`) or use the `varargin` to plot the points.

(xi) The output of the function is the time elapsed as determined by `toc`.

What to do in `hw6Ober`:

(a) Have a link to the function file on your webpage and email this file as directed.

(b) Play the game for $n = 50$ with no other inputs. Use `axis off` and `axis equal` commands. Put a meaningful title on the figure that has the number of iterations and time elapsed (using `sprintf`).

(c) Do the same as above but for $n = 1000$.

(d) Do the same as above but for $n = 50{,}000$ and specifying a black period with MarkerSize of 2.

2. **Fractal Snowflake** We will create a function `snowflakeOber` that will have a **nested function** `snowflakepoints` that will draw the Snowflake fractal at the n-th iteration. The input for `snowflakeOber` will be a nonnegative integer n (return an appropriate error message if n is not a nonnegative integer). Any additional outputs will be plot specifications. The output will be the time elapsed using the `tic` and `toc` commands.

   Pseudocode for function `snowflakeOber.m`:

   - Capture the start time.
   - Check that n is a nonnegative integer; if not, give appropriate error message using the `error` command.
   - Set $\theta = \pi/3$;
   - Create a vector v of vertices of a hexagon; calculated by $\cos(k\theta) + i\sin(k\theta)$ for $k = 1, \ldots 7$ (so they connect) for the appropriate θ. The triangle must be so the bottom base is horizontal. NOTE: you can use your `circle2Ober` function for this if you have it working properly!
   - If $n = 0$, plot the hexagon. Calculate the time elapsed and use the `return` command to end the function.
   - Initialize a vector S to be the first vertex of the hexagon (stored in v) and create a loop that will use the **nested function** `snowflakepoints` on the last element of S (the next vertex of the hexagon stored in v) for n iterations. If we looped through using the nested function `snowflakepoints` on each of the vertices of the hexagon, there would be repeated points. Thus for each of the sides of the hexagon, we replace S with S minus the last element and concatenate it with the points from the nested function `snowflakepoints` using the last element of S and the next vertex (stored in v) using the given n.

- Plot the sets of vertices S, using `varargin` for the optional inputs if needed.
- Capture the time elapsed for the output of the function.

Pseudocode for **nested function** `snowflakepoints`: has inputs a, b, and n and output vector V.

- Define the output V to have the elements a and b.
- Repeat the following n times:
  - Define an empty vector B;
  - From 1 to [(number of elements in V)-1]
    - let a = the current element of V and b = the next element of V.
    - Calculate $x1$ to be the point that is 1/3 of the way from a towards b.
    - Calculate $x2$ to be the point that is the (midpoint between a and b) PLUS $\left(\frac{i\sqrt{3}}{2}\right)\left(\frac{(b-a)}{3}\right)$
    - Calculate $x3$ to be the point 2/3 of the way from a towards b.
      (OR USE ANY DERIVATIONS FOR $x1$, $x2$, and $x3$ we did in class or you come up with on your own.) As long as it gives the proper picture, it is fine. Keep in mind that your derivation should have as few calculations as possible in order to speed up the code. To speed up the code even more, if you're using the same calculations in several of these formulas, perform the calculation first, storing it as some variable, and then use the variable within the calculations.
    - Concatenate B with the elements a, $x1$, $x2$, and $x3$.
  - Now we redefine V. The new V will be B concatenated with the last element of the current V.

(a) Have a link to `snowflakeOber` and email this file as directed.

(b) Use `subplot` to have 2 rows, 2 columns of the snowflake plotted for $n = 0, 1, 2, 3$ (capture the time elapsed for each). Hint: it may be easiest to have a `for` loop for this.

- For each snowflake/subplot, capture the time elapsed.
- For each snowflake/subplot, use the `axis equal` and `axis off` commands.
- For each snowflake/subplot, create a title with "$n =$ ___, y s" where y is the time elapsed to create that subplot. For example, you may see a title such as "$n = 3$, 0.3456 s" because the third iteration of the snowflake took 0.3456 seconds (made up numbers).

(c) Create another plot (NOT A SUBPLOT) that for $n = 5$ and optional color "Deep Sky Blue" that has rgb(0, 191, 255) (remember how you need to convert to a vector for MATLAB to recognize the RGB color). Have a similar title as above that gives the number of iterations and time elapses, and also using `axis equal` and `axis off`. Set the background color to white using `set(gcf,'Color','w')`.

(d) EXTRA CREDIT (up to 4 points): figure out how to make an animated gif of the iterations looping from $n = 0$ to $n = 5$ (no title).