15-112: Introduction to Programming and Computer Science, Spring 2020

Homework 8: Graphics

Due: Tuesday, March 24, 2020 by 22:00

This assignment has 3 questions, for a total of 50 points.
To start this homework....

1. Create a folder named "week8"
2. Create a file called hw8.py and write all your code in that file.
3. When you have completed and fully tested hw8, submit hw8.py to Autolab. For this hw, you may submit as many times as you would like, but only your last submission counts.

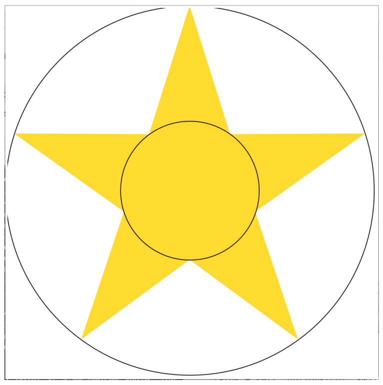
Some important notes:

1. This assignment will be **manually graded**. That means that CAs will run your code and score it appropriately. (So the autograder will not give you an instant score.)
2. Remember the course's academic integrity policy. Solving the homework yourself is your best preparation for exams and quizzes; cheating or short-cutting your learning process in order to improve your homework score will actually hurt your course grade long-term.
3. Your code will be graded for style. Check the style notes on the website for details.

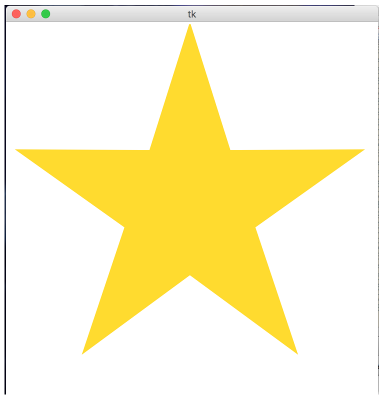
1. [10 points] `drawStar(canvas, centerX, centerY, diameter, numPoints, color)`

   Write the function drawStar which takes a canvas and the star's center coordinates, diameter, number of points, and color, and produces a star based on that specification. To draw a star, we need to identify where to place each of the inner and outer points, then draw them all together as a polygon.

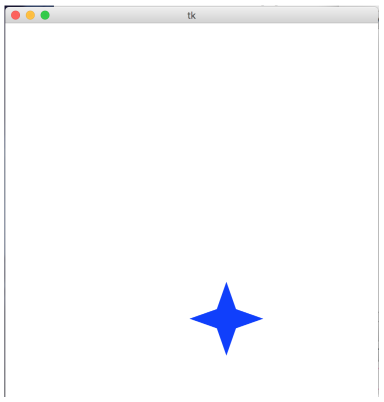
   The outer points of the star should be evenly placed on a circle based on the specified diameter, with the first point at a 90 degree angle. The inner points should then be placed on a circle 3/8 the size of the first circle, halfway between the pairs of outer points. (We use this ratio to make a nice-looking five-pointed star. Actually, the best inner circle would be about 38.2% the size of the outer circle; a little trigonometry and problem-solving will tell you why! But 3/8 is close enough.) An example of how these circles work is shown below.



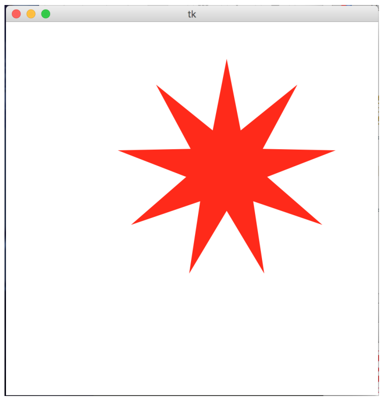
   For example, this call:
   `drawStar(canvas, 250, 250, 500, 5, "gold")`
   produces this result:



And this call:
`drawStar(canvas, 300, 400, 100, 4, "blue")`
produces this result:



And if we add a few more points:
`drawStar(canvas, 300, 200, 300, 9, "red")`
we get this result:



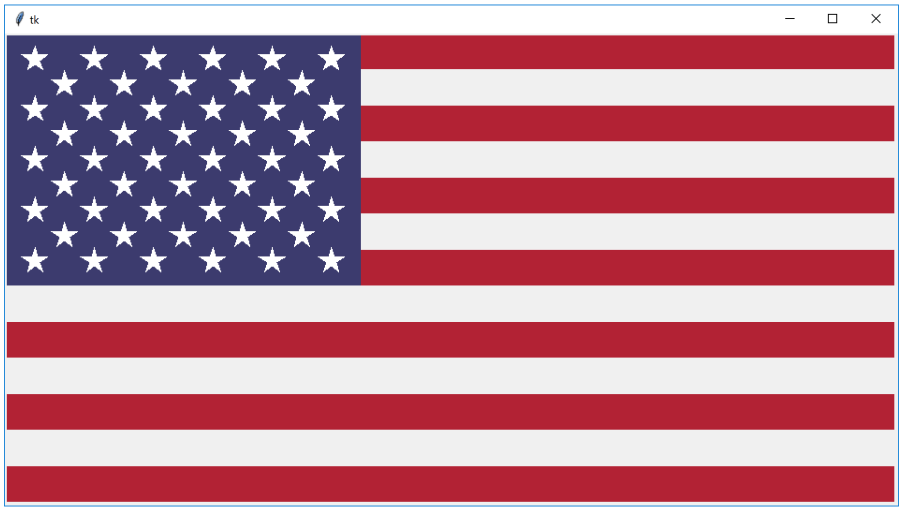
2. [20 points] `drawUnitedStatesFlag(canvas, width=950, height=500)`

   Write the function drawUnitedStatesFlag which draws the US flag in the provided dimensions. You can assume that the height:width ratio will be 10:19, as is the case with the actual US flag.

   You can find much useful information about the flag's dimensions on Wikipedia: https://en.wikipedia.org/wiki/Flag_of_the_United_States, but we do not expect you to match the actual US flag design perfectly; you should instead seek to create a reasonable approximation of the flag. However, your flag must meet the following requirements:

   1. The flag should start from the upper left-hand corner of the window.
   2. The flag should have the correct number of stripes, alternating red and white in the correct order.
   3. The blue field in the upper left corner should cover exactly seven stripes and have a reasonably correct width.
   4. The flag should have the correct number of stars in the correct configuration, with the star size and spacing reasonably close to the actual flag.
   5. The colors should be the correct shades of red and blue. (Check Wikipedia)

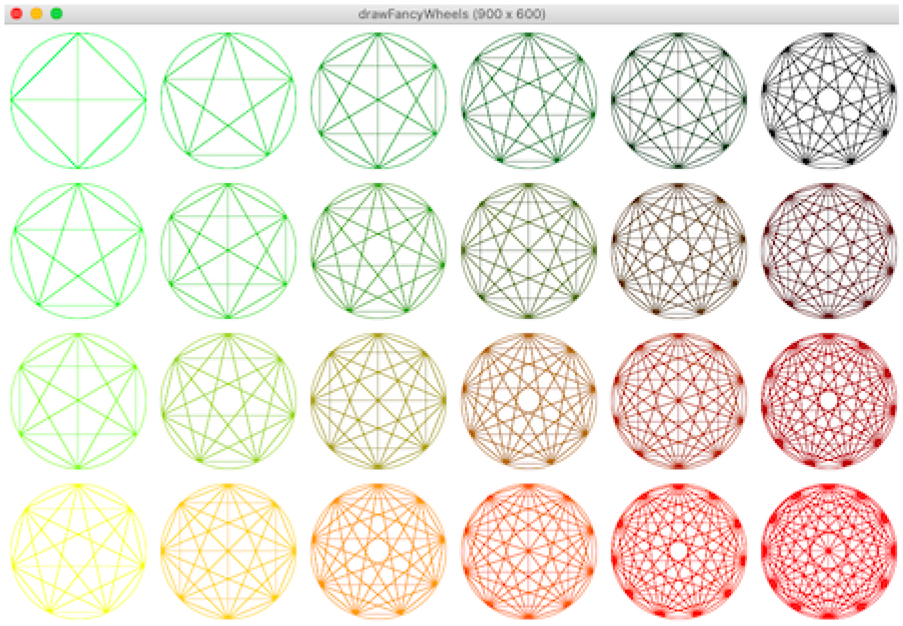
   For an example of what reasonable flags might look like, here is an example:



   Note: You should almost certainly call `drawStar` as part of solving this problem.

3. [20 points] `drawFancyWheels(canvas, width, height, rows, cols)`

   Write the function `drawFancyWheels(canvas, width, height, rows, cols)` that draws a rows-by-cols grid of so-called "fancy wheels" according to the rules below. First, though, for some context, here is the result of calling drawFancyWheels(canvas, 900, 600, 4, 6):



With that, here are the rules to follow. It may help to refer to the picture above as you go:

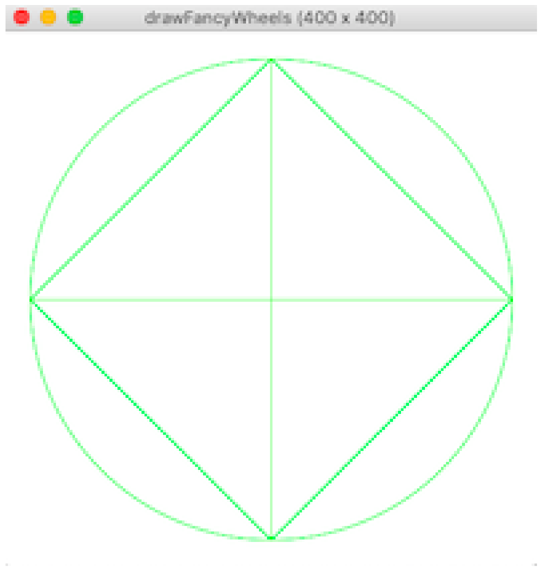
1. Think of the drawing as in a grid. Each entry in the grid is a "cell". So the drawing above is a grid with 4 rows, 6 columns, and 24 total cells.
2. The width of each cell is the total width of the canvas divided by the number of columns. The height similarly depends on the number of rows.
3. We found it very helpful to use a helper function for each cell, `drawFancyWheel(canvas, cx, cy, r, n, color)`. Here, (cx, cy) is the center of the wheel, r is its radius, n is the number of points around the wheel, and color is of course its color.
4. The radius r of each wheel depends on the smaller of the cell width and cell height. Divide this by 2 to convert from diameter to radius. Then, multiply it by 90% so the wheels do not quite entirely fill each cell.
5. The number of points n in each wheel depends on the row and the column of the wheel. The top-left wheel should have 4 points, then each wheel on the next diagonal (heading to the up-right) has 1 more point than the wheels on the previous diagonal. Look closely at the picture above to help clarify this.
6. The color of each wheel is made up of red, green, and blue components, like so:

- The red component is 0 (entirely off) in the top row, and 255 (entirely on) in the bottom row, and varies linearly in between.
- The green component is 255 in the left column, and 0 in the right column, and varies linearly in between.
- The blue component is always 0.

Note that you may find the rgbString function in the course notes to be helpful here. Also, note that the bottom-left wheel is yellow, because the RGB value of yellow is (255, 255, 0), and the top-right wheel is black, because the RGB value of black is (0, 0, 0).

7. A wheel is made up of only two kinds of shapes — a circle and some lines.
    - Each wheel includes one circle, which is drawn using the radius r of the wheel.
    - Each wheel contains n points, but these points are not ever drawn. They are only used as endpoints for the lines. That said, one point is always straight up (at 90 degrees, mathematically), and the rest are evenly distributed around the wheel.
    - To draw a wheel, draw the circle and then draw one line for each pair of points around the wheel. Thus, each wheel includes n*(n-1)//2 lines. So if n==4 for example, the wheel include 4*3//2==6 lines.
8. As you resize the graphics window, the size of the wheels should adjust, but everything else remains the same — same number of rows and columns, same colors, and so on.

Finally, to be clear, here is the result of calling `drawFancyWheels(canvas, 400, 600, 1, 1)`:


drawFancyWheels (400 x 400)