CS1112 Spring 2016 Project 4 Part A  due Thursday 4/7 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 your own, seek help from the course staff.

Objectives

Completing Part A of this project will help you learn about 2-dimensional arrays (matrices) and how they typically interact with 1-dimensional arrays (vectors). Other themes include rgb computation and the use of subfunctions.

1 Color Wheels

The display of color spectra is discussed in §4.2 in Insight. Color wheels are another approach for communicating the range of rgb possibilities:

This project is about computing and displaying color wheels¹.

A color wheel is a collection of trapezoidal radial tiles each of which is colored according to a rule. It has sectors and rings. Here is an example in which \( n_{\text{rings}} = 6 \) and \( n_{\text{sectors}} = 18 \):

¹Strictly speaking, our problem is about what are called color circles. Color wheels (used by designers) are typically based on the primary colors that apply when mixing paint: red, yellow, and blue. We work with red, green and blue.
The notation (2,11) refers to the radial tile that is in ring 2 and sector 11. To specify the location of radial tile (r,s) we define two key parameters. These are the sector angle $\Delta_{\text{sectors}} = \frac{2\pi}{n_{\text{sectors}}}$ and the ring width $\Delta_{\text{rings}} = \frac{1}{n_{\text{rings}}}$. Tile (r,s) has its two “inner” vertices on the circle with radius $(r-1)\Delta_{\text{rings}}$ and its two “outer” vertices on the circle with radius $r\Delta_{\text{rings}}$. (We are centering the color wheel at (0,0).) Notice that the tiles in ring 1 are degenerate trapezoids (a.k.a. triangles) because the inner radius is zero. The two polar angles associated with radial tile (r,s) are $(s-1)\Delta_{\text{sectors}}$ and $s\Delta_{\text{sectors}}$.

A handy way to encode the rgb values of the colors that show up in the wheel is to have a triplet of matrices $R$, $G$, and $B$, each of which is $n_{\text{rings}}$-by-$n_{\text{sectors}}$ in dimension. Our convention throughout is to house the red, green, and blue values for radial tile (r,s) in $R(r,s)$, $G(r,s)$, and $B(r,s)$ respectively.

### 1.1 Displaying a Color Wheel

Complete the following function so that it performs as specified:

```matlab
function drawWheel(R,G,B)

% Draw a color wheel defined by rgb matrices R, G, and B in the current
% figure window. Assume that the hold toggle is on.
% R, G, and B are matrices of the same size storing the red, green, and
% blue values, respectively, of the color wheel. The number of rows is the
% number of rings in the color wheel; the number of columns is the number
% of sectors in the color wheel.

% fill(x, y, c) colors the polygon defined by vectors x and y in a
% color specified by the rgb 3-vector c. The polygon will have vertices at
% (x(1), y(1)), (x(2), y(2)), ..., (x(n), y(n)) where n is the length of the
% vector x (and y). To “turn off” the default behavior that encloses
% the polygon with a black border, use the additional arguments 'LineStyle', 'none', like this:
% fill(x, y, c, 'LineStyle', 'none')

% Example script showDrawWheel that calls your function to draw a random color wheel is available on the website for your convenience.
```

For full credit, your implementation of `drawWheel` must include and make effective use of a subfunction `drawRadialTile` that draws a single radial tile. You are free to design this subfunction any way you want. Be sure to document your subfunction appropriately.

A note about using `fill`. The command `fill(x, y, c)` colors the polygon defined by vectors x and y in a color specified by the rgb 3-vector c. The polygon will have vertices at $(x(1), y(1)), (x(2), y(2)), \ldots, (x(n), y(n))$ where n is the length of the vector x (and y). To “turn off” the default behavior that encloses the polygon with a black border, use the additional arguments `LineStyle`, `none`, like this:

`fill(x, y, c, 'LineStyle', 'none')`

An example script `showDrawWheel` that calls your function to draw a random color wheel is available on the website for your convenience.

### 1.2 Shifting a Color Wheel

We can produce spirals by systematically rotating the rings of a color wheel:

For the spiral on the left, we start with the “aligned” color wheel displayed on page 1 in which $n_{\text{rings}} = 6$ and $n_{\text{sectors}} = 18$. We then rotate its rings as follows:
Instead of these single-sector “offsets,” we could have rotated the rings by other amounts. Sticking with the same example, if \( p \) is a nonnegative integer, then here is how we would generate a color wheel with offset \( p \):

For your information, the above spiral on the right has \( n_{\text{rings}} = 20 \), \( n_{\text{sectors}} = 120 \), and offset \( p = 5 \).

If \( R \), \( G \), and \( B \) are matrices that define the red, green, and blue values of an aligned color wheel, then by shifting their rows we can obtain a new triplet of matrices \( \tilde{R} \), \( \tilde{G} \), and \( \tilde{B} \) that define a spiral. Here is an illustration of how get \( \tilde{R} \) from \( R \) in the case \( n_{\text{rings}} = 5 \), \( n_{\text{sectors}} = 8 \), and \( p = 3 \):

\[
R = \begin{bmatrix}
R(1,1) & R(1,2) & R(1,3) & R(1,4) & R(1,5) & R(1,6) & R(1,7) & R(1,8) \\
R(2,1) & R(2,2) & R(2,3) & R(2,4) & R(2,5) & R(2,6) & R(2,7) & R(2,8) \\
R(3,1) & R(3,2) & R(3,3) & R(3,4) & R(3,5) & R(3,6) & R(3,7) & R(3,8) \\
R(4,1) & R(4,2) & R(4,3) & R(4,4) & R(4,5) & R(4,6) & R(4,7) & R(4,8) \\
R(5,1) & R(5,2) & R(5,3) & R(5,4) & R(5,5) & R(5,6) & R(5,7) & R(5,8)
\end{bmatrix}
\]

\[
\tilde{R} = \begin{bmatrix}
R(1,5) & R(1,6) & R(1,7) & R(1,8) & R(1,1) & R(1,2) & R(1,3) & R(1,4) \\
R(2,8) & R(2,1) & R(2,2) & R(2,3) & R(2,4) & R(2,5) & R(2,6) & R(2,7) \\
R(3,3) & R(3,4) & R(3,5) & R(3,6) & R(3,7) & R(3,8) & R(3,1) & R(3,2) \\
R(4,6) & R(4,7) & R(4,8) & R(4,1) & R(4,2) & R(4,3) & R(4,4) & R(4,5) \\
R(5,1) & R(5,2) & R(5,3) & R(5,4) & R(5,5) & R(5,6) & R(5,7) & R(5,8)
\end{bmatrix}
\]

In this example,

- You get \( \tilde{R}(1,:) \) by right-shifting \( R(1,:) \) 12 components.
- You get \( \tilde{R}(2,:) \) by right-shifting \( R(2,:) \) 9 components.
- You get \( \tilde{R}(3,:) \) by right-shifting \( R(3,:) \) 6 components.
- You get \( \tilde{R}(4,:) \) by right-shifting \( R(4,:) \) 3 components.
- You get \( \tilde{R}(5,:) \) by right-shifting \( R(5,:) \) 0 components.

The matrices \( \tilde{G} \) and \( \tilde{B} \) are computed similarly. Complete the following function so that it performs as specified.

```matlab
function [Rtilde,Gtilde,Btilde] = makeSpiral(R,G,B,p)
% R, G, and B are matrices of the same size that define a color wheel.
% p is a nonnegative integer.
% Rtilde, Gtilde, and Btilde are matrices of the same size that define a
% new color wheel that is the given color wheel with offset p.
```

<table>
<thead>
<tr>
<th>Ring</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>rotated 0 sectors counter-clockwise</td>
</tr>
<tr>
<td>5</td>
<td>rotated 1 sectors counter-clockwise</td>
</tr>
<tr>
<td>4</td>
<td>rotated 2 sectors counter-clockwise</td>
</tr>
<tr>
<td>3</td>
<td>rotated 3 sectors counter-clockwise</td>
</tr>
<tr>
<td>2</td>
<td>rotated 4 sectors counter-clockwise</td>
</tr>
<tr>
<td>1</td>
<td>rotated 5 sectors counter-clockwise</td>
</tr>
</tbody>
</table>
Your implementation must include and make effective use of a subfunction \( v = \text{shift}(u, q) \) that takes a row vector \( u \) and shifts its component values right by the amount specified by \( q \) (a nonnegative integer). Thus,

\[
v = \text{shift}([10 20 30 40 50], 3)
\]

should give \( v = [30 40 50 10 20] \). Below we introduce “modular arithmetic” and the built-in function \( \text{mod} \), which will be useful for “shifting” the components in a vector.

An example script \texttt{showMakeSpiral} is available on the course website for calling your function \texttt{makeSpiral}.

1.2.1 Modular Arithmetic

Modular Arithmetic describes a system of arithmetic for integers that “wrap around,” the most familiar of which is probably the 12-hour clock. In the 12-hour clock system, there’s no 13 o’clock or higher because the value “wraps around” at 12 and starts over at 1. 10 o’clock + 5 hours gives 3 o’clock, not 15. Arithmetic on the 12-hour clock is therefore \textit{modulo} 12, commonly read as \textit{mod} 12. Notice that 12 is congruent not only to 12 but also to 0.

To compute \( 10 + 5 \) \textit{modulo} 12, we can use the built-in function \( \text{mod} \): \( \text{mod}(10+5, 12) \) returns 3. Given positive integers \( a \) and \( b \), \( \text{mod}(a, b) \) always returns an integer in \([0..b-1]\) \footnote{Notice any similarity between \texttt{rem} and \texttt{mod}? In fact, when the arguments are positive values there’s no difference between \texttt{rem} and \texttt{mod}. Read MATLAB documentation on those two functions if you want to learn more.}. In your computation, beware of 0 if you’re dealing with vector indices! Although modular arithmetic is defined for integers, \( \text{mod} \) does work with non-integer arguments.

1.3 Making a Color Wheel

We now turn our attention to the production of the matrices \( R \), \( G \), and \( B \) that collectively define the rgb values assigned to each radial tile. We illustrate the main ideas with the example \( n_{\text{rings}} = 6 \), \( n_{\text{sectors}} = 18 \).

The first step is to determine the rgb values of the tiles that are on the “rim” of the wheel:

Notice how the colors transition from red \((s = 1)\) to yellow \((s = 4)\) to green \((s = 7)\) to cyan \((s = 10)\) to blue \((s = 13)\) to magenta \((s = 16)\) and on back to red \((s = 1)\) as we travel counterclockwise around the rim. The actual rgb values are determined by carefully sampling a function \( f \) that we shall refer to as the \textit{rim function}. The rim function \( f \) is defined on the interval \([0,6]\) as follows:

\[
f(t) = \begin{cases} 
0 & \text{if } 0 \leq t \leq 2 \\ 
t - 2 & \text{if } 2 \leq t \leq 3 \\ 
1 & \text{if } 3 \leq t \leq 5 \\ 
6 - t & \text{if } 5 \leq t \leq 6 
\end{cases}
\]
From the above plots and table we see that the blue values for the 18 rim tiles are obtained by evaluating \( f \) at 18 equally spaced points, including 0 but not 6. Similarly, the green values are obtained by sampling \( f(\text{mod}(t + 2, 6)) \) while the red values are computed by sampling \( f(\text{mod}(t + 4, 6)) \).

Once the color of a rim tile is known, then we can compute the color of the other tiles in its sector by using the following “darkening” rule:

\[
\text{If } c \text{ is the rgb vector that defines the color of the rim tile in sector } s, \text{ then the color of tile } (r, s) \text{ is given by } \rho \cdot c \text{ where the value of } \rho \text{ is } \sin\left(\frac{r}{n_{\text{rings}}} \frac{\pi}{2}\right) \]

Notice that the value of \( \sin\left(\frac{r}{n_{\text{rings}}} \frac{\pi}{2}\right) \) decreases as the ring index \( r \) decreases forcing the tiles to get darker as we move towards the center of the wheel.

Now that we have described how the rim tiles and interior tiles are colored, it is possible for you to complete the following function so that it performs as specified:

```matlab
function [R,G,B] = makeWheel(nRings,nSectors)
% nRings is a positive integer
% nSectors is a positive integer >= 3.
% R, G, and B are matrices with nRings rows and nSectors columns
% that define a color wheel that has nRings rings and nSectors
% sectors. The rgb color of radial tile (r,s) is given by
% R(r,s), G(r,s), and B(r,s).

% Include a subfunction to return the value of the rim function described above given t.

% An example script showMakeMakeWheel is available on the course website for calling your function makeWheel.

% Play with the parameters in showMakeMakeWheel to create different spirals!

% Submit your files drawWheel.m, makeSpiral.m, and makeWheel.m on CMS.

Project 4 Part B will appear in a separate document.
```