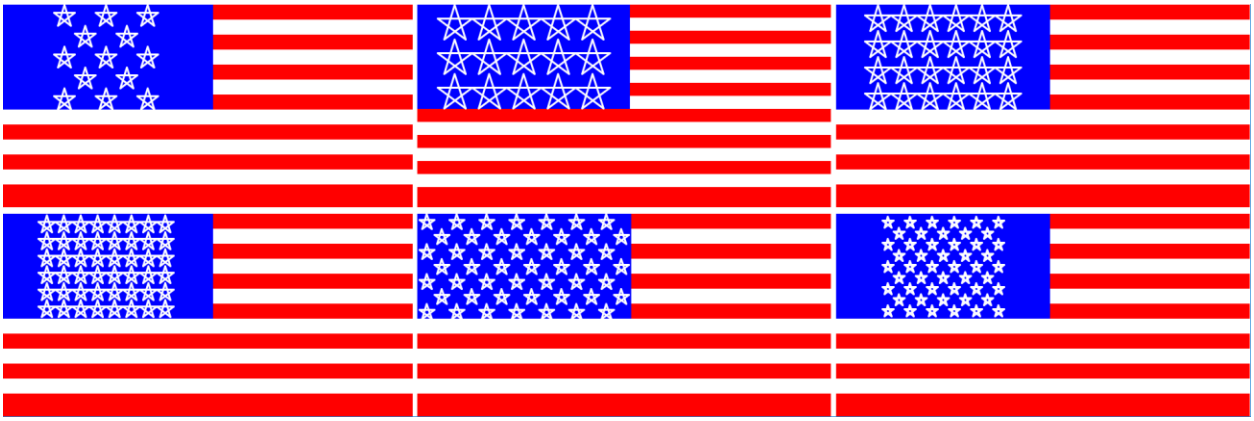
CS 142 Assignment 3
Stars and Stripes Forever
Please see Canvas for the due date!

In this assignment, you will **write a class** `StarsAndStripes` that can draw a generalized United States flag. The following image represents some of the kinds of flags that your program may be able to draw:



Write the following method which draws a US flag representation with the given number of stars and stripes starting at the given (x, y) coordinate (upper left corner in Java's coordinate system) and of the given width and height. Notice that a flag may fill only part of the window like the six flags drawn above!

```
public static void drawFlag(int stars, int stripes, java.awt.Graphics g,
                            int x, int y, int width, int height)
```

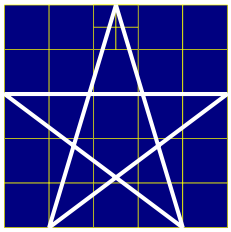
Assume stars, stripes, width, and height are at least one. You should only call fillRect, drawLine, and setColor methods on the java.awt.Graphics object. Color.red, Color.white, Color.blue are the only colors allowed. To reduce confusion and differences in calculation, you should only perform math with **`int`**, not with **`double`**. **In addition, another method is specified on the back of this worksheet.**

Your `drawFlag` method should follow the following procedure so that it can be properly tested:

First, draw a filled white rectangle (*base*) of the given x, y, width, and height.

Second, draw filled red rectangles (*red stripes*) across the width of the *base*. The first rectangle should begin in the *base*'s upper left. Each *red stripe* height (except possibly the last one) should be (`height` divided by `stripes`) **rounded down**. Then, skip down according to the *red stripe* height which will expose the *base*. Draw a number of *red stripes* equal to half of `stripes` **rounded up**. If `stripes` is odd, the final *red stripe* height should touch the bottom pixels of the underlying *base* rectangle, not beyond.

Third, draw a filled blue rectangle (*starfield*). The *starfield*'s height should be equal to the *red stripe* height times the number of *red stripes*. Given this height, the *starfield*'s width should be proportional to the flag (*starfield* height × *base* width / *base* height). You should **round down** for the width calculation.



Finally, draw the white stars. Each star will have its own upper left (x, y) as well as an equal width and height and will be drawn as line segments resembling the diagram to the left: from the bottom 1/5 of the way from the left to 2/5 of the way from the top on the right, straight across to the left, down to the bottom 1/5 of the way from the right, up to the top center, and back down to the starting point.

Each calculation to draw the star line coordinates should round down. You should implement the following helper method which draws a single star and call it from your drawFlag method when it needs to draw stars. **Note that size denotes both star width and star height since they are equal.**

```
public static void drawStar(java.awt.Graphics g, int x, int y, int size)
```

This will allow the test program to separately test your star drawing procedure and make sure that it works correctly. Once you know how to draw a star using the above method, you will have to place a bunch of them on your flag! Your program will have to plan out a strategy to figure out where the stars should be drawn.

Determine whether the stars can be drawn in a rectangular grid whose number of columns is greater than the number of rows but less than two times the number of rows (the 6×4, 8×6, 5×3 flags as depicted; note that the historical 15 star flag didn't actually use this arrangement). Given the number of stars as an argument, your code should search for all possible ways to divide it up. If so, draw the stars in the rectangular grid pattern of stars inside the starfield. When drawing stars, each star should be the same size, the stars should be as large as they can be within their grid while still fitting in the starfield, and if there is leftover space, try to center the stars within the starfield. Note that due to line thickness, the edges of the lines may go beyond the starfield, but the lines coordinates should stay within it.

Otherwise, for **extra credit**, if the above grid strategy doesn't work, you'll have to draw the stars in a checkerboard grid. First, try to consider an odd-by-odd checkerboard of stars where the columns is equal to the rows or two more than the rows and the upper left corner is in use: this is the case for the 13 star flag (5 by 5 checkerboard of stars) and the 50 star flag (11 by 9 checkerboard of stars). To receive extra credit, your code should not hardcode the 13 and 50 star flag arrangements but should find them algorithmically using this approach, which should work for other sizes.

For **additional extra credit**, after the above, consider any possible checkerboard arrangement where the number of columns is greater than or equal to the number of rows but less than three times the number of rows. You should prefer odd numbers of rows and columns and prefer using the upper left corner of the checkerboard. The bottom center flag on page 1 is the arrangement of stars on the 49 star flag.

Here are examples of 51, 52, 55, 58, 59, and 61 star staggered flags with increasing numbers of stripes: