EEN118 LAB TWO

The purpose of this lab is to get practice with defining and using your own functions. The essence of good structured programming is to split large problems into smaller and smaller sub-problems. Small sub-problems should have small solutions which can be programmed quickly and reliably. Functions that solve each of the small sub-problems are used together to solve the original big problem. Divide and conquer.

If you just go immediately to the last step and wonder how to do it, this may seem like an impossibly long and detailed assignment. If you work through the numbered steps properly, you will soon see that it isn't.

After each step, look over what you have done carefully, and make sure you completely understand it, and that it does not seem particularly complicated any more. When starting the next step, think carefully how you can use what you have already done to make it easy.

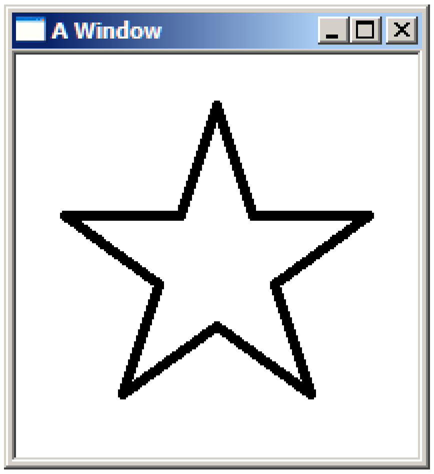
1. *A Five-Pointed Star.*

In a previous lab exercise, you wrote a program that draws a pentagram, which is almost a five-pointed star, but has a pentagon inscribed inside it. Adapt that program so that it draws a clean five-pointed star with no extra lines, just like you would expect to see on an American flag.

There are many ways to solve a problem in programming. This one has two equally sensible solutions. You could write a function `void draw_star(const int length)`, where length is the length of each of the ten lines that make up the star. The programmer would be responsible for moving the pen to the right position before calling this function.

The other would be to write `void draw_star(const int x, const int y, const int length)`, which required the programmer to work out the numeric x-y coordinates of the top spike of the star before calling the function.

Think ahead. Think about how you are going to be using this function, and make what you judge to be the appropriate decision.

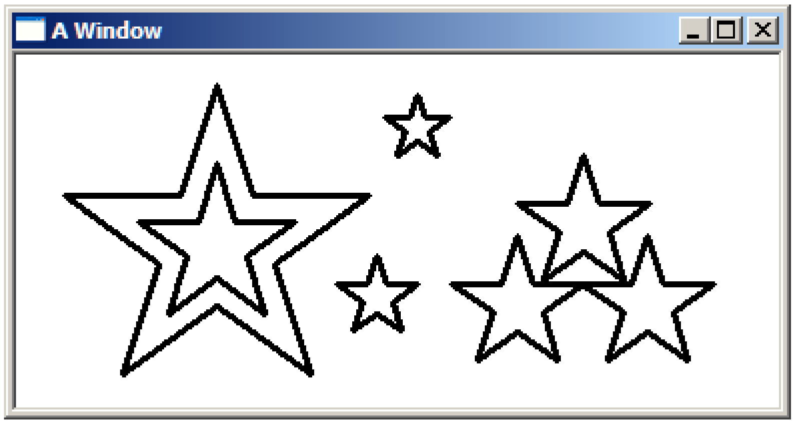


You should be able to work out what the correct angles are, but in case you can't, they are given above the first diagram on the next page.

2. *A Function.*

Convert the part of your program that actually draws the star, after all the set-up, into a function. Make this function good and flexible (it will need to have at least one parameter), so that it can draw a star of any size and at any position.
Verify that you have got it right by using your function to draw a variety of stars in the same window. Make sure they all come out with the right orientation.

The angles inside the points of the star are 36°, the wider "shoulder" angles are 108°.



3. *Colouring It In.*

Drawing the outline of a star is all very well, but they would look much better if filled with some solid colour, or perhaps if they were white shapes on a coloured background.

Filling a whole rectangular area with a single colour is easy, and there is a library function that does it in a single operation. Just give it the x and y co-ordinates of the top left corner of the area, then the width, then the height, and it's done. Naturally, it uses whatever the current pen colour is, e.g.:

```
set_pen_color(color::blue);
fill_rectangle(10, 10, 200, 50);
```

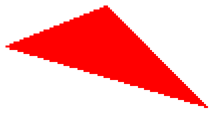
will fill the whole area from x=10 to x=210 and y=10 to y=60 in blue.

To fill a more complex shape, your program must define each of its corners one by one. There are three simple steps:

a. Use the `start_shape()` function, just to tell the system that a new shape is about to be defined.
b. Move the pen to each of the corners in turn. When the pen is positioned at a corner, use the `note_position()` function to tell the system to record this position as a point on the outline of the shape. You don't need to draw, just move the pen to the right places.
c. Finally, set the pen to the right colour, and call the `fill_shape()` function.

Here is a simple example that draws a slightly irregular solid red triangle:

```
start_shape();
move_to(100, 100);
note_position();
move_to(150, 150);
note_position();
move_to(50, 120);
note_position();
set_pen_color(color::red);
fill_shape();
```

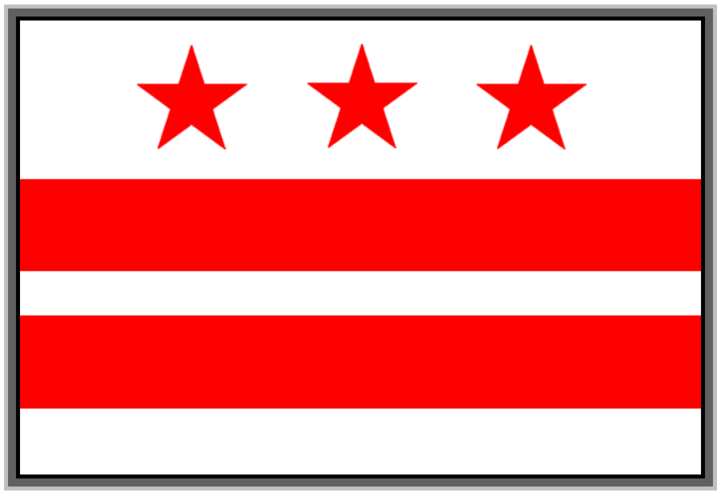


It is a peculiarity of windows that whenever it fills a shape, it also draws the outline using the current pen settings. If you haven't selected a very thin pen, the shape will have a slightly blurry or rounded appearance.

Make your star function draw solid sharp stars.

4. *An Intermediate Flag.*

Test your function in a program that draws the flag of Washington D.C. It will just take a minute or two of experimentation to get the proportions looking exactly right. Don't be afraid of trial-and-error. This is how it should look:

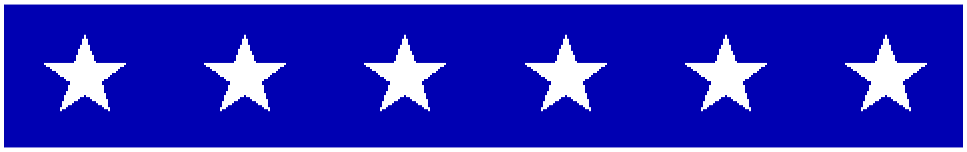


5. *Rows of Stars.*

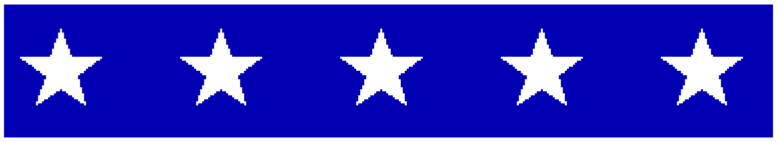
Write a new function that makes use of your existing star-drawing function to draw a horizontal row of six evenly spaced solid white stars. If you have got the right idea, this will be very easy.

You have already completely solved the problem of drawing one solid white star. There is no need to modify that function in any way. All you have to do for this new function is to call the existing function a few times.

Give the new function a sensible name, something like row_of_six_stars, and make sure that it is good and flexible: it should be able to draw its row of stars anywhere you want.



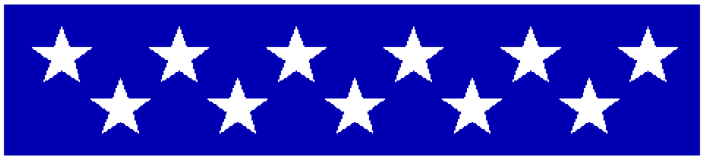
Once your row of six stars looks right, take a few seconds to make another function which this time draws a row of just five stars.



At this stage, you should have two new functions, one that draws a row of six stars, and one that draws a row of five stars.

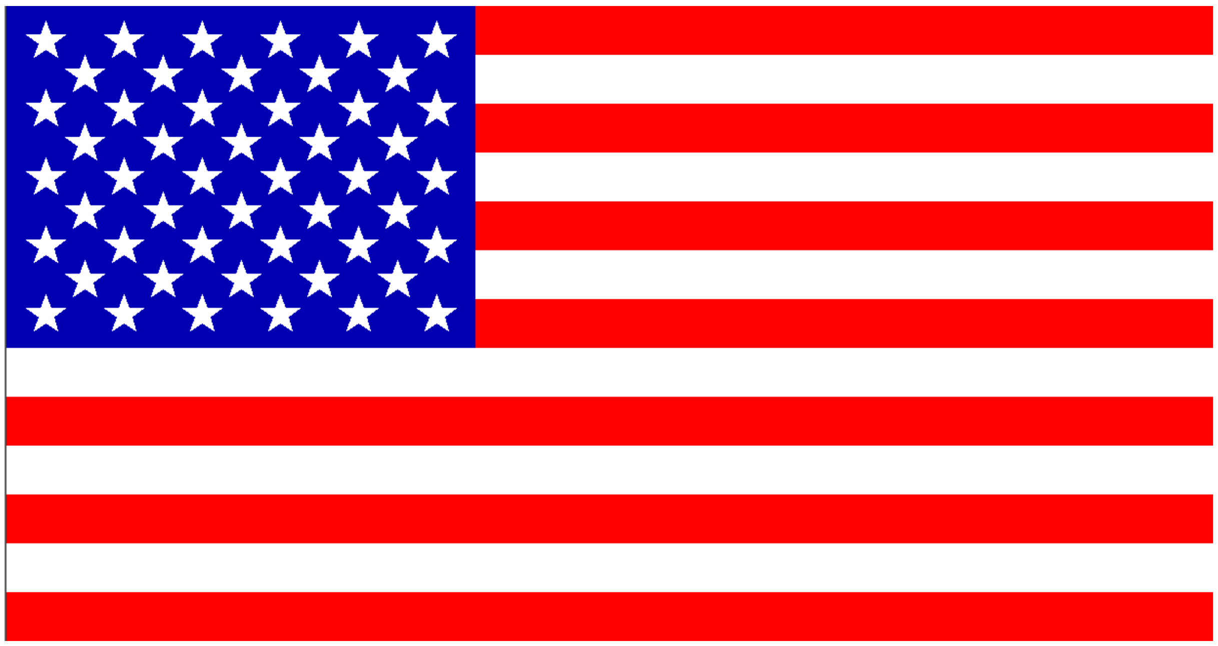
6. *Blocks of Stars.*

Now it will take almost no effort to write another function that calls both your `row_of_six_stars` and `row_of_five_stars` functions to make a block of eleven stars:



Take the time to get the stars properly spaced, so that the two rows slightly overlap as shown, because you know what is coming next. Get this stage right, and the next will not be very hard at all. Make this into a new function too, drawing eleven stars just like that.

You are now at the point where your solutions to smaller sub-problems can be made into a relatively simple solution to a much larger and more complex problem. Write a program that draws a good rendering of the American flag.



Try to get it looking just right. After all this effort, you might as well do it properly. Officially the ratio of width to height of the flag should be 19 to 10.

As you can tell by counting stripes, the height of the blue box is seven thirteenths of the height of the whole flag. The length of the blue box is two fifths of the length of the whole flag.

8. *Extra Credit: Make it Special.*

What size should it be? We know the correct proportions, but flags are allowed to be any overall size you want. Make your program flexible: define one constant called `height`, which defines the height of the whole flag in pixels, and calculate all other dimensions from that. It should be that all you have to do is change the value of `height`, and all the rest of the flag will automatically adjust to continue looking just right at the new size.