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 decomposing a large problem into a number of smaller sub-problems (and those sub-problems into even smaller sub-sub-problems, and so on and so on, as far as you need to go). Each of the sub-problems is solved by functions which the programmer must design, implement, and tie together into a single program.

Correctly identifying sub-problems is an important task. Whenever you think you have identified a sub-problem that can be solved with a function that will contribute to the solution of the whole problem, make sure that you can describe that sub-problem in a short simple sentence or two. If a task hasn't got a nice simple description, then maybe it isn't a very good task: maybe it is just part of a larger task, or maybe it needs to be split up into yet smaller tasks.

Having a clear description of the exact requirements of a sub-problem is equivalent to having a clear idea of what the function you're about to write must do. If the description of what a function must do is complicated or long, then the function itself will turn out to be complicated and long, and therefore unreliable.

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.

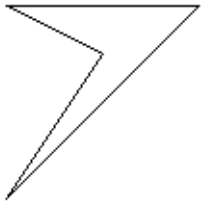
2. *A Function.*

   Convert the star-drawing part of your program into a function which you could use many times to draw as many five-pointed stars as you like. The function should be *position independent*, which means that you could use it to draw a star anywhere in the graphics window, not just in one fixed place. The function should also have a parameter that defines the size of the star (by providing the length of the ten lines that make up the outline of the star). To save you from too much geometry, the angles are shown on the last page of this lab. Use your function to draw a series of stars of different sizes and in various places.

3. Colouring In

The graphics library contains a collection of functions that can record the outline of a shape as you are drawing it, and then fill it in with a solid colour. There are three important functions to use: `StartShape()` tells the system to start recording a new shape, `AddHereToShape()` tells the system to record the current pen position as one of the vertices (corners) of the shape being recorded, and `FillShape()` tells the system to fill in the outline just recorded with solid colour. You don't *need* to actually draw the outline: you can just move the pen around on the window, tracing out the outline of the shape, while recording the position of each corner. Here is an example; the following piece of program would just draw an irregular red quadrilateral:

```
PenColor(PEN_RED);
MoveTo(100, 100);
DrawLineTo(200, 100);
DrawLineTo(100, 200);
DrawLineTo(150, 125);
DrawLineTo(100, 100);
```



Simply adding in calls to the new functions produces a solidly coloured version of the same shape:

```
PenColor(PEN_RED);
StartShape();
MoveTo(100, 100);
AddHereToShape();
DrawLineTo(200, 100);
AddHereToShape();
DrawLineTo(100, 200);
AddHereToShape();
DrawLineTo(150, 125);
AddHereToShape();
DrawLineTo(100, 100);
AddHereToShape();
FillShape();
```

Use this new method to modify your star-drawing function so that it draws solid stars, not just outlines.

4. The Lone Star State

Incorporate your function into a program that draws the flag of Texas as nicely as possible.

There is another function that makes the drawing of solid rectangles easier:

```
FillRectangleXYWH(x,y,w,h)
```

draws a solid rectangle whose top left corner is at coordinates (x,y), with a width of w pixels and a height of h pixels. Or you could just draw a line with a very wide pen.

5. *Two More Simple Functions*

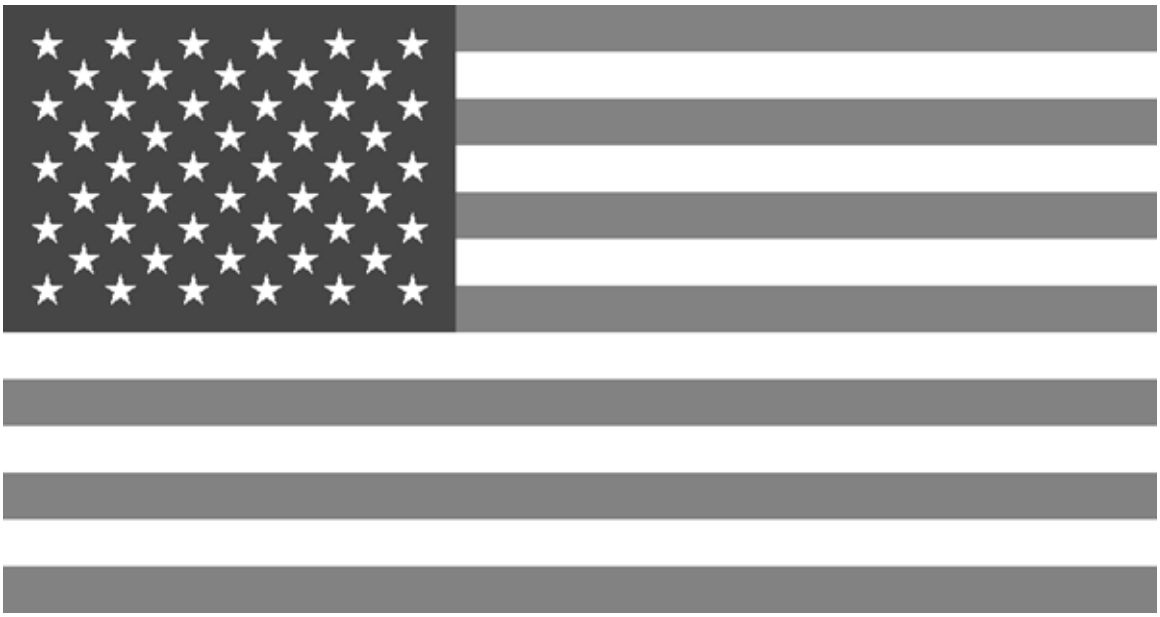
Using the solid-star-drawing function you have already written (but you can forget about the Texan flag), write one function that draws an evenly spaced row of six stars, and another function that draws an evenly spaced row of five stars. Make them solid white stars on a blue background.

Then use your two functions together to produce a nicely aligned double row of eleven stars:

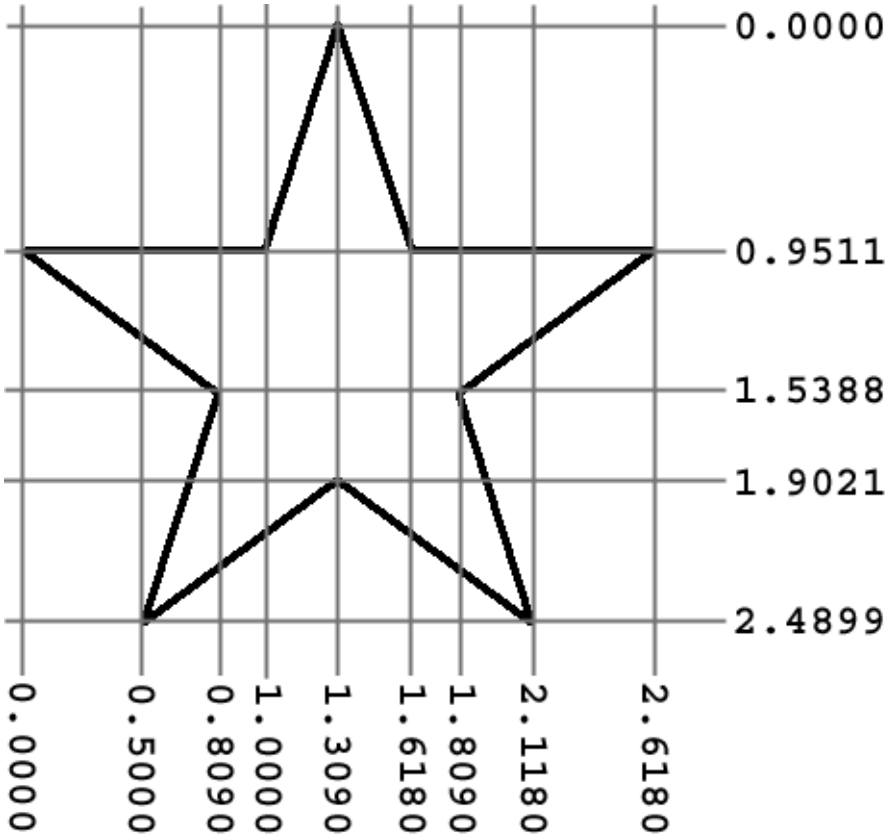
6. *You Have Probably Guessed What's Coming Next...*

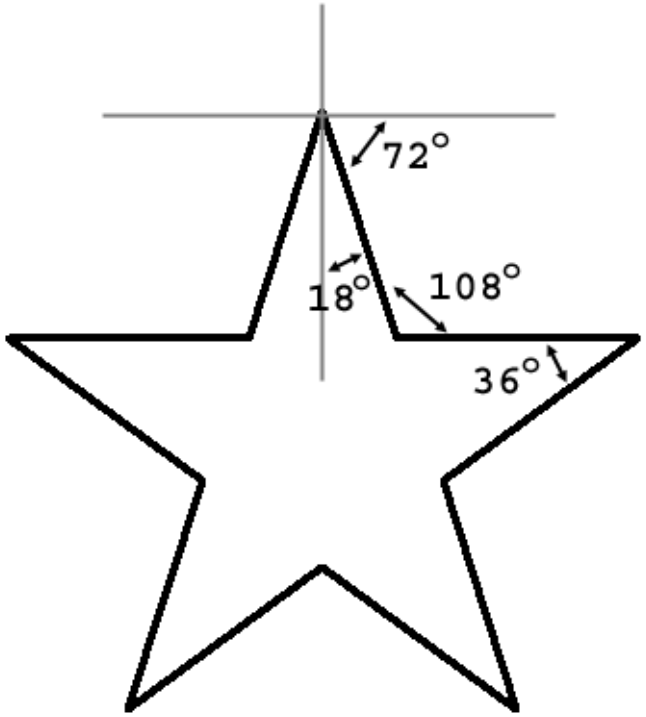
Write a program that draws an accurate rendering of an American Flag. The program must make good use of functions so that it remains readable and understandable even though it is performing quite a complex task.

This picture shows the correct configuration of the stars and stripes on the flag. You don't have to copy it slavishly, but try to make your program produce good-looking accurate results. The official proportions of the flag (length to height) are 19 to 10.



The Geometry of a five-pointed star.





Your Own Design

A projectile shot straight upwards from a cannon with a muzzle velocity of v feet per second rises to a maximum height then crashes back to earth after a total of $2v/g$ seconds in the air (where g is the "acceleration due to gravity", 32 feet per second per second). The height of the projectile above ground at any time can be calculated by this formula:

$$h = vt - \tfrac{1}{2}gt^2$$

in which t is the time (in seconds) since the cannon was fired.

Write a program that prints the altitude of such a projectile at times 1, 2, 3, 4, ..., up to 10 seconds into its flight, for some value of v. You will have to edit the program to make it work for a different value of v; don't worry about trying to make it interactive.

If it seems that your program is going to be a little messy, you are right, but don't worry about it. I would expect the formula to appear 10 times in your program. Soon you will see new techniques that let you avoid such things. For now, let the editor do most of the work (cut and paste), and type carefully. You won't appreciate good techniques if you have never suffered from the problems that they prevent.

Then, use the graphics functions to plot a line graph showing the trajectory instead of just printing it as a table of numbers. The vertical axis should be altitude in feet, and the horizontal should be time in seconds. You will need to allow more than one pixel per second on the horizontal axis if you are to be able to see the plot properly.