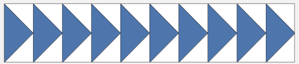
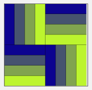
Lab 8

- #1-3 – “work with any size window”
  - use `width`, `height` instead of specific numbers as appropriate
  - be careful to cover all applicable places
  - test by changing the values in `size()`

Lab 8

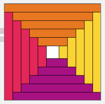


- #1 – flying geese
  - “triangle should span full height of window”
    - triangle’s height should be `height`
  - “triangle’s height is twice its width”
    - triangle’s width is `height/2`
  - “repeat as many times as fully fits, leaving space on the right if there isn’t room for a whole triangle”
    - repeat-as-long-as pattern
    - check the tip of the rectangle against the right side of the window (`width`)



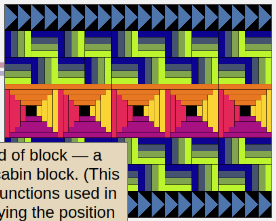
- #2 – rail fence
  - “stripes sized so four stripes exactly fill each quadrant”
    - vertical stripes are `height/2` tall, `width/8` wide
    - horizontal stripes are `width/2` wide, `height/8` tall
    - counting loop pattern

Lab 8



- #3
  - getting the right pattern of rectangles – note carefully how they mesh with each other
    - what changes from one repetition to the next? position (x,y) and size (width or height)
  - “rectangles should be 20 pixels tall/wide”
    - short side should be 20, long side’s initial value should be based on height or width
  - “include as many rectangles as fit, leaving space in the middle if there isn’t enough room for an entire set of rectangles”
    - repeat-as-long-as pattern
    - should always have the same number of each color (but resulting from the repeat-as-long-as condition, not a counting loop)
  - “rectangle needs to fit entirely within its half of the window”
    - compare right edge (red) or left edge (yellow) position to `width/2` or bottom edge (orange) or top edge (purple) position to `height/2`

Lab 8



- #4

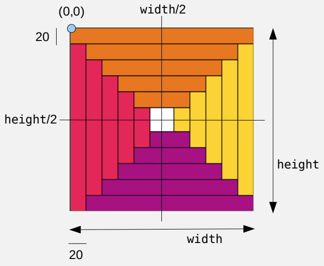
Your sketch should contain a drawing function for each kind of block — a strip of flying geese, a single rail fence block, a single log cabin block. (This means three drawing functions, in addition to the drawing functions used in #2.) Each drawing function should have parameters specifying the position and size of the block, and should draw the pattern filling that region. Don't forget to include comments describing each function and its parameters.

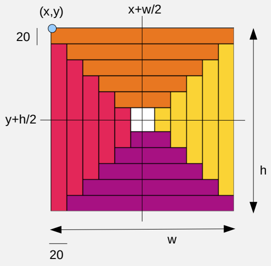
  - choose descriptive names for the functions
    - `drawFlyingGeese`, `drawRailFence`, `drawLogCabin`
  - `drawRailFence` and `drawLogCabin` should each draw a single block
    - use a loop to repeat the block 5 times for one row in the quilt
  - each drawing function should have parameters for position and size of the block
    - x, y, w, h

Lab 8

Writing the body of each function should largely be a matter of copying the relevant code from #1-3, with two changes: within each function, the parameters and loop variables must have different names (so you might need to rename some things), and the pattern now needs to be drawn within the region specified by the parameters rather than filling the whole drawing window.

- be careful to fully adjust from block-fills-drawing-window to block-with-position-and-size-specified-by-parameters



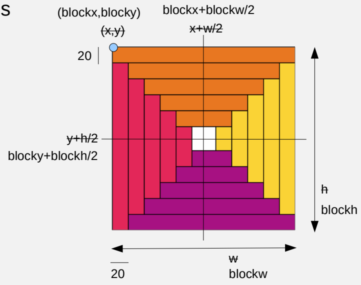


Writing the body of each function should largely be a matter of copying the relevant code from #1-3, with two changes: within each function, the parameters and loop variables must have different names (so you might need to rename some things), and the pattern now needs to be drawn within the region specified by the parameters rather than filling the whole drawing window.

- can't have both function parameters x, y, w, h (for the position and size of the block) and loop variables x, y, w, h (for position and size changing from one rectangle to the next)

  → e.g. use blockx, blocky, blockw, blockh for the function parameters

  (or rename the loop variables instead)


(blockx,blocky)
~~(x,y)~~
blockx+blockw/2
~~x+w/2~~
20
~~y+h/2~~
blocky+blockh/2
~~h~~
blockh
~~w~~
blockw
20