Problem 7:

We will use Transformation matrix that we change a Current (x, y) Position into the next (xt, yt)

begin = Vec([x, y, 1])

Matrix = Matrix(...)

next = Start * matrix / multiplication of matrices

now go backwards to figure out the matrix if your next is expected to look like Vec([a*x+x`, b*y+y`, 1]).

Examine only the x component at first. we'll Take the dot Product of our start Vector and the topmost row of our matrix, resulting in a*x+x`

we can do the same thing for the other two rows of the matrix to get the following:

Matrix

$$\begin{bmatrix} [a, 0, x`] \\ [0, b, y`] \\ [1, 0, 0] \end{bmatrix}$$

If we Take our start vector and multiply it with this matrix, we'll get the translated vector. ~~next~~ ~~as Vec~~

The actual beauty of this is that the matrix doesn't Care what our inputs are at all it's fully ~~Self contain~~ Self-Contained.

Fortunately, we have an efficient method for computing a matrix to a power it's called exponentiation by squaring.

In a nutshell, rather than multiplying the number or matrix n times we square it and multiply intermediate Vale by original number / matrix ~~at the~~ at the appropriate intervals. quickly In log(n) Time