STATISTICAL RETHINKING 2023
WEEK 1 SOLUTIONS

1. You can use grid approximation or just the Beta distribution directly. I'll show both. The grad approximation can reuse the code from the chapter. But I'll rewrite it a bit so the function accepts the counts W and L instead:

```
compute_posterior <- function( W , L , poss=c(0,0.25,0.5,0.75,1) ) {
    ways <- sapply( poss , function(q) q^W * (1-q)^L )
    post <- ways/sum(ways)
    data.frame( poss , ways , post=round(post,3) )
}
compute_posterior( 4 , 11 , poss=seq(from=0,to=1,len=11) )
```

```
   poss         ways  post
1   0.0 0.000000e+00 0.000
2   0.1 3.138106e-05 0.068
3   0.2 1.374390e-04 0.300
4   0.3 1.601635e-04 0.350
5   0.4 9.287605e-05 0.203
6   0.5 3.051758e-05 0.067
7   0.6 5.435818e-06 0.012
8   0.7 4.253299e-07 0.001
9   0.8 8.388608e-09 0.000
10  0.9 6.561000e-12 0.000
11  1.0 0.000000e+00 0.000
```

Using the Beta distribution means we don't really need to compute the distribution. We have a mathematical expression for it. There's nothing to compute. But you can plot it with:

```
curve( dbeta(x,4+1,11+1) , from=0 , to=1 , xlab="p" )
```

2. Let's sample from the Beta distribution and then simulate globe tosses from those samples:

```
p_samples <- rbeta(1e4,4+1,11+1)
W_sim <- rbinom(1e4,size=5,p=p_samples)
```

I used the rbinom() function, but you could use sample() and then tally the water points. The resulting distribution is approximated by the counts in W_sim. You can view the distribution with:

```
plot(table(W_sim))
```

3. All we need to do is count how many samples in W_sim satisfy the criterion. So we ask:

```
sum( W_sim >= 3 )
```

```
1819
```

You'll get a slightly different answer, because of simulation variance. But the point is that there are 1e4 simulations, so if 1819 are 3 or greater, then the probability of 3 or greater is approximately 0.18.

4 - CHALLENGE. The first insight needed here is to define a sequence for N rather than for p. Then the same code almost works. Only almost because N has a known lower bound—it is at least W. And it has no defined upper bound. The globe could in principle be tossed an infinite number of times. Not practically but mathematically. So our posterior function needs to know how large an N we'd like to consider.

The second insight is that unlike the book example, the sequence of W and L isn't known here. So we have to consider how many different sequences could produce any particular mix of W and L. Luckily the binomial distribution does this for us. I'll make the calculation explicit, but you could just use dbinom().

```
compute_posterior_N <- function( W , p , N_max ) {
    ways <- sapply( W:N_max ,
        function(n) choose(n,W) * p^W * (1-p)^(n-W) )
    post <- ways/sum(ways)
    data.frame( N=W:N_max , ways , post=round(post,3) )
}
compute_posterior_N( W=5 , p=0.7 , N_max=20 )
```

```
    N         ways  post
1   5 1.680700e-01 0.118
2   6 3.025260e-01 0.212
3   7 3.176523e-01 0.222
4   8 2.541218e-01 0.178
5   9 1.715322e-01 0.120
6  10 1.029193e-01 0.072
```

```
7  11 5.660564e-02 0.040
8  12 2.911147e-02 0.020
9  13 1.419184e-02 0.010
10 14 6.622860e-03 0.005
11 15 2.980287e-03 0.002
12 16 1.300489e-03 0.001
13 17 5.527078e-04 0.000
14 18 2.295863e-04 0.000
15 19 9.347442e-05 0.000
16 20 3.738977e-05 0.000
```

Since p is greater than 0.5, we expect most tosses to be W, so the posterior distribution for N assigns most of the probability to values close to the observed $W = 5$. If we make p small, we'll get the opposite:

```
compute_posterior_N( W=5 , p=0.2 , N_max=20 )
```

```
    N       ways  post
1   5 0.00032000 0.000
2   6 0.00153600 0.001
3   7 0.00430080 0.004
4   8 0.00917504 0.008
5   9 0.01651507 0.014
6  10 0.02642412 0.023
7  11 0.03875537 0.034
8  12 0.05315022 0.046
9  13 0.06909529 0.060
10 14 0.08598525 0.075
11 15 0.10318229 0.089
12 16 0.12006667 0.104
13 17 0.13607556 0.118
14 18 0.15072985 0.131
15 19 0.16364955 0.142
16 20 0.17455952 0.151
```

If you had any prior information about N, you could add that to the function as well. Just multiply. For example, suppose you recall that you always toss the globe an even number of times. Then we could just zero out the odd numbers and renormalize.