Exercise 1.19. There is a clever algorithm for computing the Fibonacci numbers in a logarithmic number of steps. Recall the transformation of the state variables a and b in the `fib-iter` process of section 1.2.2: $a \leftarrow a+b$ and $b \leftarrow a$. Call this transformation T, and observe that applying T over and over again n times, starting with 1 and 0, produces the pair Fib $(n+1)$ and Fib (n). In other words, the Fibonacci numbers are produced by applying T^n, the nth power of the transformation T, starting with the pair $(1, 0)$. Now consider T to be the special case of $p=0$ and $q=1$ in a family of transformations T_{pq}, where T_{pq} transforms the pair (a, b) according to $a \leftarrow b\,q + a\,q + a\,p$ and $b \leftarrow b\,p + a\,q$. Show that if we apply such a transformation T_{pq} twice, the effect is the same as using a single transformation $T_{p'q'}$ of the same form, and compute p' and q' in terms of p and q. This gives us an explicit way to square these transformations, and thus we can compute T^n using successive squaring, as in the `fast-expt` procedure. Put this all together to complete the following procedure, which runs in a logarithmic number of steps:[1]

```
(define (fib n)
  (fib-iter 1 0 0 1 n))
(define (fib-iter a b p q count)
  (cond ((= count 0) b)
        ((even? count)
         (fib-iter a
                   b
                   <??>    ; compute p'
                   <??>    ; compute q'
                   (/ count 2)))
        (else (fib-iter (+ (* b q) (* a q) (* a p))
                        (+ (* b p) (* a q))
                        p
                        q
                        (- count 1)))))
```

Answer. By reading the description above, we may easily conclude that the key point to solving this problem relys on how to represent p' and q' in terms of p and q. With observation, we may immediately draw out our basic strategy in a straightforwardly: apply the transformation T_{pq} to the pair (a, b) twice, then compare the consequence with that of the single transformation $T_{p'q'}$.

In order to present the intermediate result in a clear way, we first denote the original pair (a, b) by (a_0, b_0), that is,

$$a_0 = a, b_0 = b$$

Well, Let's see how the pair (a, b) involves while being transformed by T_{pq} for the first time:

$$\begin{aligned} a_1 &= b_0\,q + a_0\,q + a_0\,p \\ &= b\,q + a\,q + a\,p \\ &= b\,q + a\,(p+q) \end{aligned}$$

$$\begin{aligned} b_1 &= b_0\,p + a_0\,q \\ &= b\,p + a\,q \end{aligned}$$

$$\begin{cases} a_1 = b\,q + a\,(p+q) \\ b_1 = b\,p + a\,q \end{cases}$$

Continue in this way, the value of pair (a_2, b_2) can be obtained gradually as follow:

$$\begin{aligned} a_2 &= b_1\,q + a_1\,(p+q) \\ &= (b\,p + a\,q)\,q + (b\,q + a\,(p+q))(p+q) \\ &= b\,p\,q + a\,q^2 + b\,p\,q + b\,q^2 + a\,(p^2 + 2\,p\,q + q^2) \\ &= b\,(q^2 + 2\,p\,q) + a\,(p^2 + q^2 + q^2 + 2\,p\,q) \end{aligned}$$

$$\begin{aligned} b_2 &= b_1\,p + a_1\,q \\ &= (b\,p + a\,q)\,p + (b\,q + a\,(p+q))\,q \\ &= b\,p^2 + a\,p\,q + b\,q^2 + a\,(p\,q + q^2) \\ &= b\,(p^2 + q^2) + a\,(q^2 + 2\,p\,q) \end{aligned}$$

$$\begin{cases} a_2 = b\,(q^2 + 2\,p\,q) + a\,(p^2 + q^2 + q^2 + 2\,p\,q) \\ b_2 = b\,(p^2 + q^2) + a\,(q^2 + 2\,p\,q) \end{cases}$$

1. This exercise was suggested to us by Joe Stoy, based on an example in Kaldewaij 1990.

By Comparing the expression of the pair (a_2, b_2) above with that of the single transformation $T_{p'q'}$ where

$$\begin{cases} a_2 = b\,q' + a\,(p' + q') \\ b_2 = b\,p' + a\,q' \end{cases}$$

we can find out the expression of p' and q' in terms of p and q intuitively:

$$\begin{cases} p' = p^2 + q^2 \\ q' = q^2 + 2\,p\,q \end{cases}$$

Finally, we have to complete the `fib` procedure to bring an end to our solution:

```
(define (fib n)
  (fib-iter 1 0 0 1 n))
(define (fib-iter a b p q count)
  (cond ((= count 0) b)
        ((even? count)
         (fib-iter a
                   b
                   (sum-of-squares p q)        ; compute p'
                   (+ (square q) (* 2 p q))    ; compute q'
                   (/ count 2)))
        (else (fib-iter (+ (* b q) (* a q) (* a p))
                        (+ (* b p) (* a q))
                        p
                        q
                        (- count 1)))))
```