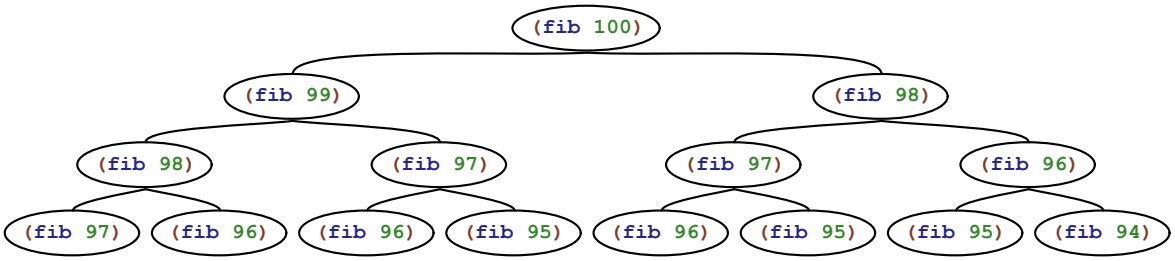
```
; The Fibonacci Sequence and Combinatorial Explosion

; There's a sequence of numbers called "The Fibonacci Sequence".

; It has many applications. If you're interested, see:
;   https://en.wikipedia.org/wiki/Fibonacci_number
;  and Vi Hart's videos:
;   https://www.youtube.com/watch?v=ahXIMUkSXX0

; Start with: 1 1
; Add the previous two numbers, to get the next number.
;         1   (fib 0)
;         1   (fib 1)
; 1 + 1 = 2   (fib 2)
; 1 + 2 = 3   (fib 3)
; 2 + 3 = 5   (fib 4)
; 3 + 5 = 8   (fib 5)
; 5 + 8 = 13  (fib 6)
; 8 + 13 = 21 (fib 7)
; ...

; Let's make a function ‘fib’
;   fib :  number  →  number
; (fib n) produces the nth number in the Fibonacci Sequence.

(check-expect (fib 0) 1)
(check-expect (fib 1) 1)

(check-expect (fib 7) 21)
(check-expect (fib 7) (+ 8 13))
(check-expect (fib 7) (+ (fib 5) (fib 6)))
(check-expect (fib 7) (+ (fib (- 7 2)) (fib (- 7 1))))

; Design:
(check-expect (fib 0) 1)
(check-expect (fib 1) 1)
(check-expect (fib 7) (+ (fib (- 7 2)) (fib (- 7 1))))

(define (fib n)
  (cond [(= n 0) 1]
        [(= n 1) 1]
        [else (+ (fib (- n 2))
                 (fib (- n 1)))]))

; Steps for (fib 2):
(fib 2)
;
(+ (fib 0)
   (fib 1))
;
(+ 1
   1)
;
2
```

```
; Steps for (fib 3):
(fib 3)
;
(+ (fib 1)
   (fib 2))
;
; WE know that is:
(+ 1
   2)
;
; But the COMPUTER does this:
(+ 1 ; This came from (fib 1)
   (+ (fib 0)
      (fib 1))) ; This calculates (fib 1) again.
;
(+ 1
   (+ 1
      1))
;
(+ 1 2)
;
3

; Steps for (fib 4):
(fib 4)
(+ (fib 2)
   (fib 3))
(+ (+ (fib 0)
      (fib 1))
   (+ (fib 1) ; Re-calculation of (fib 1).
      (fib 2))) ; Re-calculation if (fib 2).
; ...
(+ (+ 1 1)
   (+ 1 2))
(+ 2 3)
5

; Steps for (fib 35):
(fib 35)
;
(+ (fib 33)
   (fib 34))
;
(+ (+ (fib 31)
      (fib 32))
   (+ (fib 32)
      (fib 33)))
;
(+ (+ (fib 31)
      (fib 32))
   (+ (fib 32)
      (+ (fib 31)
         (fib 32))))
;
; It asks for (fib 32) three times there.
; It asks for (fib 29) three times three times: nine times.
; (fib 26) will be asked 27 times from that.
; (fib 23) will be asked 81 times ....
```

```
; Accumulating multiplication.
; (expt b n) is b raised to the power of n.
; In other words: (* b b b ...) n times.
(check-expect (expt 10 3) (* 10 10 10))
(check-expect (expt 10 3) 1000)
(check-expect (expt 10 6) (* 10 10 10 10 10 10))
(check-expect (expt 10 6) 1000000)

(check-expect (expt 2 1) (* 2))
(check-expect (expt 2 1) 2)
(check-expect (expt 2 2) (* 2 2))
(check-expect (expt 2 2) 4)
(check-expect (expt 2 3) (* 2 2 2))
(check-expect (expt 2 3) 8)

; Part of the computation of (fib 100):
```




```
; That's almost a doubling of work about 100 times:
(check-expect (expt 2 100) 1267650600228229401496703205376)
; So don't try (fib 100)!
```