

```
1 #lang racket
2
3 (struct Zero () #:transparent)
4 (struct Succ (n) #:transparent)
5
6 (define (make-number prev)
7   (match prev
8     ((Succ prev) (Succ (Succ prev)))
9     ((Zero) (Succ (Zero)))
10    (_ (printf "Error: not a valid number!"))))
11
12 (define one (make-number (Zero)))
13 one
14
15 (define two (make-number one))
16
17 two
18
19 (define (addition n1 n2)
20   (match n1
21     ((Zero) n2)
22     ((Succ i)(addition i (Succ n2)))))
23
24 (addition two two)
25
26 (struct Pred (n) #:transparent)
27
28 (Pred (Zero))
29
30 (define (make-number-2 prev)
31   (match prev
32     ((Succ prev) (Succ (Succ prev)))
33     ((Zero) (Succ (Zero)))
34     ((Pred n) n)
35     (_ (printf "Error: not a valid number!\n"))))
36
37 (make-number-2 (Zero))
38
```