Concurso Auxiliar de 2da categoria

Programacion I

Damian Ariel Marotte

Enunciado

Diseñe la funcion `sumanat` que toma dos *numeros naturales* y sin usar `+` devuelve un *natural* que es la suma de ambos.

Tenga en cuenta la definicion dada en clase de los *numeros naturales*, sus constructores, y predicados para la definicion de la funcion pedida.

Numeros naturales

Un *Natural* es:

- 0
- `(add1 Natural)`

Un *Natural* es:

- `0`
- `(add1 Natural)`

Predicados:

- `zero?` `; Reconoce al natural 0`
- `positive?` `; Reconoce naturales construidos con add1`

Un *Natural* es:

- `0`
- `(add1 Natural)`

Predicados:

- `zero?` *; Reconoce al natural 0*
- `positive?` *; Reconoce naturales construidos con add1*

Selector:

- `sub1` *; Devuelve el predecesor de un numero natural*

Receta

1. Diseño de datos
2. Signatura
3. Declaracion de proposito
4. Ejemplos
5. Codigo
6. Testing
7. Correccion

Diseño de datos

```
;n: Natural (Representa el primer sumando)
;m: Natural (Representa el segundo sumando)
```

Signatura

```
1  ;n: Natural (Representa el primer sumando)
2  ;m: Natural (Representa el segundo sumando)
3
4
5
6
7
8  ;sumanat : Natural Natural -> Natural
9
10
11
12
13
14
15
16
17
18
19
```

Declaracion de proposito

```
;n: Natural (Representa el primer sumando)
;m: Natural (Representa el segundo sumando)




;sumanat : Natural Natural -> Natural
;Dados los numeros naturales n y m, devuelve el numero natural n + m
```

Ejemplos

```
1  ;n: Natural (Representa el primer sumando)
2  ;m: Natural (Representa el segundo sumando)
3  ;(sumanat 0 0) = 1 (Ambos 0)
4  ;(sumanat 0 3) = 3 (Primero 0)
5  ;(sumanat 7 0) = 7 (Segundo 0)
6  ;(sumanat 2 3) = 5 (Ninguno 0)
7  ;(sumanat 3 2) = 5 (Conmutatividad)
8  ;sumanat : Natural Natural -> Natural
9  ;Dados los numeros naturales n y m, devuelve el numero natural n + m
10
11
12
13
14
15
16
17
18
19
```

Codigo

```
;n: Natural (Representa el primer sumando)
;m: Natural (Representa el segundo sumando)
;(sumanat 0 0) = 1 (Ambos 0)
;(sumanat 0 3) = 3 (Primero 0)
;(sumanat 7 0) = 7 (Segundo 0)
;(sumanat 2 3) = 5 (Ninguno 0)
;(sumanat 3 2) = 5 (Conmutatividad)
;sumanat : Natural Natural -> Natural
;Dados los numeros naturales n y m, devuelve el numero natural n + m
(define (sumanat n m)



)
```

Codigo

```
;n: Natural (Representa el primer sumando)
;m: Natural (Representa el segundo sumando)
;(sumanat 0 0) = 1 (Ambos 0)
;(sumanat 0 3) = 3 (Primero 0)
;(sumanat 7 0) = 7 (Segundo 0)
;(sumanat 2 3) = 5 (Ninguno 0)
;(sumanat 3 2) = 5 (Conmutatividad)
;sumanat : Natural Natural -> Natural
;Dados los numeros naturales n y m, devuelve el numero natural n + m
(define (sumanat n m)
    (cond [(zero? n)                                    ] ; caso base
          [(positive? n)                                ] ; caso recursivo
    )
)
```

Codigo

```
;n: Natural (Representa el primer sumando)
;m: Natural (Representa el segundo sumando)
;(sumanat 0 0) = 1 (Ambos 0)
;(sumanat 0 3) = 3 (Primero 0)
;(sumanat 7 0) = 7 (Segundo 0)
;(sumanat 2 3) = 5 (Ninguno 0)
;(sumanat 3 2) = 5 (Conmutatividad)
;sumanat : Natural Natural -> Natural
;Dados los numeros naturales n y m, devuelve el numero natural n + m
(define (sumanat n m)
    (cond [(zero? n)      m                          ] ; caso base
          [(positive? n)                             ] ; caso recursivo
    )
)
```

Codigo

```
;n: Natural (Representa el primer sumando)
;m: Natural (Representa el segundo sumando)
;(sumanat 0 0) = 1 (Ambos 0)
;(sumanat 0 3) = 3 (Primero 0)
;(sumanat 7 0) = 7 (Segundo 0)
;(sumanat 2 3) = 5 (Ninguno 0)
;(sumanat 3 2) = 5 (Conmutatividad)
;sumanat : Natural Natural -> Natural
;Dados los numeros naturales n y m, devuelve el numero natural n + m
(define (sumanat n m)
    (cond [(zero? n)      m                          ] ; caso base
          [(positive? n) (sumanat n m)              ] ; caso recursivo
    )
)
```

Codigo

```
1  ;n: Natural (Representa el primer sumando)
2  ;m: Natural (Representa el segundo sumando)
3  ;(sumanat 0 0) = 1 (Ambos 0)
4  ;(sumanat 0 3) = 3 (Primero 0)
5  ;(sumanat 7 0) = 7 (Segundo 0)
6  ;(sumanat 2 3) = 5 (Ninguno 0)
7  ;(sumanat 3 2) = 5 (Conmutatividad)
8  ;sumanat : Natural Natural -> Natural
9  ;Dados los numeros naturales n y m, devuelve el numero natural n + m
10 (define (sumanat n m)
11     (cond [(zero? n)      m                          ] ; caso base
12           [(positive? n) (sumanat (sub1 n) m)       ] ; caso recursivo
13     )
14 )
15
16
17
18
19
```

Codigo

```
1  ;n: Natural (Representa el primer sumando)
2  ;m: Natural (Representa el segundo sumando)
3  ;(sumanat 0 0) = 1 (Ambos 0)
4  ;(sumanat 0 3) = 3 (Primero 0)
5  ;(sumanat 7 0) = 7 (Segundo 0)
6  ;(sumanat 2 3) = 5 (Ninguno 0)
7  ;(sumanat 3 2) = 5 (Conmutatividad)
8  ;sumanat : Natural Natural -> Natural
9  ;Dados los numeros naturales n y m, devuelve el numero natural n + m
10 (define (sumanat n m)
11     (cond [(zero? n)      m                          ] ; caso base
12           [(positive? n) (sumanat (sub1 n) (add1 m))] ; caso recursivo
13     )
14 )
15
16
17
18
19
```

Codigo

```
;n: Natural (Representa el primer sumando)
;m: Natural (Representa el segundo sumando)
;(sumanat 0 0) = 1 (Ambos 0)
;(sumanat 0 3) = 3 (Primero 0)
;(sumanat 7 0) = 7 (Segundo 0)
;(sumanat 2 3) = 5 (Ninguno 0)
;(sumanat 3 2) = 5 (Conmutatividad)
;sumanat : Natural Natural -> Natural
;Dados los numeros naturales n y m, devuelve el numero natural n + m
(define (sumanat n m)
    (cond [(zero? n)      m                          ] ; caso base
          [(positive? n) (sumanat (sub1 n) (add1 m))] ; caso recursivo
    )
)
(check-expect (sumanat 0 0) 0)
(check-expect (sumanat 0 3) 3)
(check-expect (sumanat 7 0) 7)
(check-expect (sumanat 2 3) 5)
(check-expect (sumanat 3 2) 9)
```

Ran 5 tests.
1 of the 5 tests failed.

No signature violations.

Check failures:
Actual value **5** differs from **9**, the expected value.
at line 19, column 0

Correccion

```
;n: Natural (Representa el primer sumando)
;m: Natural (Representa el segundo sumando)
;(sumanat 0 0) = 1 (Ambos 0)
;(sumanat 0 3) = 3 (Primero 0)
;(sumanat 7 0) = 7 (Segundo 0)
;(sumanat 2 3) = 5 (Ninguno 0)
;(sumanat 3 2) = 5 (Conmutatividad)
;sumanat : Natural Natural -> Natural
;Dados los numeros naturales n y m, devuelve el numero natural n + m
(define (sumanat n m)
    (cond [(zero? n)      m                          ] ; caso base
          [(positive? n) (sumanat (sub1 n) (add1 m))] ; caso recursivo
    )
)
(check-expect (sumanat 0 0) 0)
(check-expect (sumanat 0 3) 3)
(check-expect (sumanat 7 0) 7)
(check-expect (sumanat 2 3) 5)
(check-expect (sumanat 3 2) 9)
```

Correccion

```
;n: Natural (Representa el primer sumando)
;m: Natural (Representa el segundo sumando)
;(sumanat 0 0) = 1 (Ambos 0)
;(sumanat 0 3) = 3 (Primero 0)
;(sumanat 7 0) = 7 (Segundo 0)
;(sumanat 2 3) = 5 (Ninguno 0)
;(sumanat 3 2) = 5 (Conmutatividad)
;sumanat : Natural Natural -> Natural
;Dados los numeros naturales n y m, devuelve el numero natural n + m
(define (sumanat n m)
    (cond [(zero? n)      m                          ] ; caso base
          [(positive? n) (sumanat (sub1 n) (add1 m))] ; caso recursivo
    )
)
(check-expect (sumanat 0 0) 0)
(check-expect (sumanat 0 3) 3)
(check-expect (sumanat 7 0) 7)
(check-expect (sumanat 2 3) 5)
(check-expect (sumanat 3 2) 5)
```

All 5 test passed!