**Capítulo 9 --------------------------Octavio Reyes Pinedo**

**Exercise 1.** Is the following a chain? You can test your conclusions by evaluating

s in each case.

imp1 :: Integer -> Integer

imp1 1 = 2

imp1 x = error "imp1: premise does not apply"

imp2 :: Integer -> Integer

imp2 2 = 3

imp2 x = error "imp2: premise does not apply"

imp3 :: Integer -> Integer

imp3 3 = 4

imp3 x = error "imp3: premise does not apply"

s :: [Integer]

s = [1, imp1 (s !! 0), imp2 (s !! 1), imp3 (s !! 2)]

--------------RESPUESTA---------------

---- Si es una cadena porque se obtiene [1, 2, 3, 4]

**Exercise 2.** Is the following a chain?

imp1\_2 :: Integer -> Integer

imp1\_2 1 = 2

imp1\_2 x = error "imp1: premise does not apply"

imp2\_2 :: Integer -> Integer

imp2\_2 3 = 4

imp2\_2 x = error "imp2: premise does not apply"

s\_2 :: [Integer]

s\_2 = [0, imp1\_2 (s\_2 !! 0), imp2\_2 (s\_2 !! 1)]

----------------------RESPUESTA-----------------------------------------

----Se obtiene [0,\*\*\* Exception: imp1\_2: premise does not apply

--No es una cadena porque en imp1\_2 no se define el parametro 0, lo que causa una excepción.

**Exercise 3.** Is the following a chain?

imp1\_3 :: Integer -> Integer

imp1\_3 0 = 1

imp1\_3 x = error "imp1: premise does not apply"

imp2\_3 :: Integer -> Integer

imp2\_3 3 = 4

imp2\_3 x = error "imp2: premise does not apply"

s\_3 :: [Integer]

s\_3 = [0, imp1\_3 (s\_3 !! 0), imp2\_3 (s\_3 !! 1)]

-----------------------RESPUESTA------------------------------------------------

---Se obtiene [0,1,\*\*\* Exception: imp2: premise does not apply

--No es una cadena, porque en imp2 no se define el parametro 1, lo que causa el

--error "imp2:premise does not apply"

**Exercise 4.** Is the following a chain?

imp1\_4 :: Integer -> Integer

imp1\_4 0 = 1

imp1\_4 x = error "imp1: premise does not apply"

imp2\_4 :: Integer -> Integer

imp2\_4 1 = 2

imp2\_4 x = error "imp2: premise does not apply"

s\_4 :: [Integer]

s\_4 = [0, imp1\_4 (s\_4 !! 1), imp2\_4 (s\_4 !! 0)]

-------------RESPUESTA----------------------------------

--No es una cadena, porque en s\_4

--imp1\_4 recibe como parametro 1 en lugar de recibir 0

--además de que en imp2\_4 necesita el parametro 1, pero

--se envía el parametro 0

--En s\_4 lo correcto sería = [0, imp1\_4 (s\_4 !! 0), imp2\_4 (s\_4 !! 1)]

**Exercise 5.** Given the base case 0 *∈ n* and the induction rule *x ∈ n → x*+1 *∈*

*n*, fix the following calculation so that 3 is in set *n*:

--fun :: Integer -> Integer

--fun x = x - 1

--n :: [Integer]

--n = 0 : map fun n

--------------------------CORREGIDO-------------------

fun :: Integer -> Integer

fun x = x + 1

n :: [Integer]

n = 0 : map fun n

--Al ejecutar \*Main> 3 `elem` n

--Se obtiene True

**Exercise 6.** Use the following definitions, determine whether 4 is in set *s*,

given 1 *∈ s* and the induction rule *x ∈ s → x* + 2 *∈ s*.

fun6 :: Integer -> Integer

fun6 x = x + 2

s6 :: [Integer]

s6 = 1 : map fun6 s6

------------RESPUESTA----------------

-- Al ejecutar \*Main> s6

-- Se obtiene [1,3,5,7,9,11,13,15,17...]

-- Por lo tanto 4 no se encuentra en el conjunto (solamente los números impares)

**Exercise 7.** Fix this calculation of the positive integers:

fun :: Integer -> Integer

fun x = 0

p :: [Integer]

p = 0 : map fun p

-----------------------CORREGIDO------------------------

fun7 :: Integer -> Integer

fun7 x = x + 1

p7 :: [Integer]

p7 = 0 : map fun7 p7

-- Al ejecutar \*Main> p7

-- Se obtiene [0,1,2,3,4,5,6,7,8,9...]

**Exercise 8.** Fix this calculation of the positive multiples of 3:

fun :: Integer -> Integer

fun x = x \* 3

p :: [Integer]

p = map fun p

----------------------CORREGIDO-----------------------------

fun8 :: Integer -> Integer

fun8 x = x \* 3

p8 :: [Integer]

p8 = map fun8 [1..]

--Al ejecutar \*Main> p8

--Se obtiene [3,6,9,12,...]

**Exercise 9.** Here is a Haskell equation that defines the set s inductively. Is

82 an element of s?

s9 :: [Integer]

s9 = 0 : map ((+) 2) s9

------------------------RESPUESTA----------------------:

--Si se incluye el 82, porque se incluyen todos los numeros pares.

**Exercise 10.** What set is defined by the following?

s10 :: [Integer]

s10 = 1 : map ((\*) 3) s10

----------------RESPUESTA----------------------

-- Al ejecutar \*Main> s10

-- Se obtiene[1,3,9,27,81...]

-- que son el resultado de multiplicar cada elemento por 3 (1, 1\*3=3, 3\*3=9, 9\*3=27, 27\*3=81...)

**Exercise 11.** Alter the definition of newBinaryWords and binWords so that

they produce all of the *octal* numbers. An octal number is one that

contains only the digits 0 through 7.

--------------------------RESPUESTA--------------------------------

octals = [0..7]

newOctalWords :: [Integer] -> [[Integer]]

newOctalWords xs = [x:xs | x <- octals]

octalWords = [[x] | x <- octals] ++ (concatMap newOctalWords octalWords)

--Al ejecutar \*Main>octalWords

--Se obtiene [[0],[1],[2],[3],[4],[5],[6],[7],[0,0],[1,0],[2,0],[3,0],[4,0],[5,0],[6,0],[7,0],[0,1].......]

**Exercise 12.** Use take 10 integers1 to evaluate the first 10 integers according

to this definition. Describe the set that is actually defined by Attempt

1.

------------------------Attempt 1--------------------------------

build :: a -> (a -> a) -> [a]

build a f = set

where set = a : map f set

builds :: a -> (a -> [a]) -> [a]

builds a f = set

where set = a : concatMap f set

nextInteger1 :: Integer -> Integer

nextInteger1 x = - x

integers1 :: [Integer]

integers1 = build 0 nextInteger1

----------------------RESPUESTA-----------------------------

ex12 = take 10 integers1

--Se obtiene [0,0,0,0,0,0,0,0,0,0]

--Ya que build 0 nextInteger1

-- 0 : map nextInteger1 (0:set)

-- 0 : - 0 : map nextInteger1 (-0)

-- Entonces al ejecutar integers1 se obtiene un conjunto de solamente ceros [0,0,0,0.....]

**Exercise 13.** Use take 20 integers2 to evaluate the first 20 integers according

to this definition. Describe the set that is actually defined by Attempt

2.

------------------------Attempt 2--------------------------------

nextIntegers2 :: Integer -> [Integer]

nextIntegers2 x = [x + 1, x - 1]

integers2 :: [Integer]

integers2 = builds 0 nextIntegers2

----------------------RESPUESTA-----------------------------

ex13=take 20 integers2

--Al ejecutar \*Main> ex13

--Se obtiene [0,1,-1,2,0,0,-2,3,1,1,-1,1,-1,-1,-3,4,2,2,0,2]

-- builds 0 nextIntegers2

-- [0] ++ [1, -1] ++ concatMap nextIntegers2 ([0] ++ [1, -1] ++ ...)

-- [0,1,-1] ++ [1,-1] ++ [2, -2] ++ [3, -3] ...

-- Cada elemento es presentado multiples veces pero se busca que cada elemento sea mostrado solo una vez

**Exercise 14.** Use the computer to examine the first 10 integers generated by

this definition, and describe the set that is defined.

------------------------Attempt 3--------------------------------

nextIntegers3 :: Integer -> [Integer]

nextIntegers3 x = [x + 1, -(x + 1)]

integers3 :: [Integer]

integers3 = builds 0 nextIntegers3

-----------------------RESPUESTA-------------------------------

ex14=take 10 integers3

-- Al ejecutar \*Main>ex14

-- Se obtiene [0,1,-1,2,-2,0,0,3,-3,-1]

-- Aún se muestra cada número varias veces

-- con 2 se obtiene [3,-3]

-- con 1 se obtiene [2,-2]

-- con 0 se obtiene [1,-1]

-- con -1 se obtiene [0,-1]

-- con -2 se obtiene [-1,1]

**Exercise 15.** Use the computer to generate some elements of the set defined

by Attempt 4, and describe the result.

------------------------Attempt 4--------------------------------

nextInteger4 :: Integer -> Integer

nextInteger4 x = if x < 0 then x - 1 else x + 1

integers4 :: [Integer]

integers4 = build 0 nextInteger4

-----------------------RESPUESTA-------------------------------

ex15=take 10 integers4

--Al ejecutar \*Main>ex15

-- se obtiene [0,1,2,3,4,5,6,7,8,9]

-- Pero se siguen obteniendo solo números positivos porque por default,

-- solamente 0 está en el conjunto desde que 0 no es menor que 0

-- y se siguen agregando solo números positivos.

**Exercise 16.** Use the computer to evaluate the first 10 elements of the set,

and describe the result.

------------------------Attempt 5--------------------------------

nextIntegers5 :: Integer -> [Integer]

nextIntegers5 x = if x > 0 || x == 0

then [x + 1, -(x + 1)]

else []

integers5 :: [Integer]

integers5 = builds 0 nextIntegers5

-----------------------RESPUESTA-------------------------------

ex16=take 20 integers5

--Al ejecutar \*Main>ex16

--Se obtiene [0,1,-1,2,-2,3,-3,4,-4,5,-5,6,-6,7,-7,8,-8,9,-9,10]

-- si el elemento x es mayor o igual a cero se hace el cálculo [x+1, -(x+1)]

-- de lo contrario se queda vacío []

-- con 0 se obtiene 1,-1

-- con 1 se obtiene 2,-2

-- con -1 se obtiene []

-- con 2 se obtiene 3,-3

-- con -2 se obtiene []

-- con 3 se obtiene 4,-4

-- con -3 se obtiene []

-- así sucesivamente hasta obtener todos los enteros y sin repetirlos.

**Exercise 17.** Does ints, using the following definition, enumerate the integers?

If it does, then you should be able to pick any integer and see it

eventually in the output produced by ints. Will you ever see the value

-1?

nats :: [Integer]

nats = build 0 (1 +)

negs :: [Integer]

negs = build (-1) (1 -)

ints :: [Integer]

ints = nats ++ negs

**-----------------------RESPUESTA----------------------------------------**

-- Does this definition of ints enumerate the integers?

-- Si,porque incluye a nats que contiene todos los enteros positivos [0,1,2,3,4,5...] y también a negs que contiene [-1,2,....]

-- Will you ever see the value -1?

-- No, porque se tendrían que imprimir en pantalla TODOS los enteros positivos antes de que se pudiera visualizar en pantalla el valor de -1

**Exercise 18.** Does twos enumerate the set of even natural numbers?

twos :: [Integer]

twos = build 0 (2 \*)

-----------RESPUESTA---------------

-- No, porque twos contiene solamente ceros [0,0,0,0,0,0,0,0,0,0...]

**Exercise 19.** What is wrong with the following definition of the stream of

natural numbers?

nats9 = map (+ 1) nats9 ++ [0]

----------------RESPUESTA--------------------------------------

-- El caso base aparece al fin de la cadena, entonces este no puede ser usado

-- para iniciar el proceso inductivo de calcular los elementos sucesivos. Entonces

-- la cadena no tiene un valor que sea imprimible. Por lo tanto no muestra ningún resultado.

**Exercise 20.** What is the problem with the following definition of the naturals?

naturals :: [Integer] -> [Integer]

naturals (i:acc) = naturals (i + 1:i:acc)

nats20 :: [Integer]

nats20 = naturals [0]

-- La ejecución nunca terminará. Es decir, la ejecución realizará...

-- naturals(0:[])=

-- naturals(1:0:[])=

-- naturals(2:1:0:[])=

-- naturals(3:2:1:0:[])=....

-- Por lo tanto no tendrá fin y no aparecerá ningún resultado en pantalla.

**Exercise 21.** Can we write a function that will take a stream of the naturals

(appearing in any order) and give the index of a particular number?

**­-----------------------RESPUESTA---------------------------------**

--No, no se puede porque el stream primero tratará de generar una cantidad infinita de números

--naturales antes de poder mostrar asignar el índice al número deseado que estemos buscando.

--Por ejemplo, en [1,3..] ++ [2,4..] TODOS los números impares aparecerán antes que

--cualquier número par (que nos serviría como índice).

**Exercise 22.** Using induction, define the set of roots of a given number *n*.

**-------------------------**Respuesta**---------------------**

-- Dado un número n, el conjunto R de raíces de n es definido como sigue:

-- \* n1 *∈* R

-- \* n1/m *∈* R -> n1/m+1 *∈* R

-- \* Nada está en R a menos que pueda demostrarse que está en R por un numero finito

--de usos de la base y reglas de inducción. (Es decir, cualquiera que no pueda ser creado por

--un número finito de aplicaciones de las reglas y base no es una raíz)

**Exercise 23.** Given the following definition, prove that *n*3 is in set P of powers of *n*.

**Definition 28.** Given a number *n*, the set *P* of powers of *n* is defined

as follows:

*• n*0 *∈ P*

*• nm ∈ P → nm*+1 *∈ P*

*•* Nothing else is in *P* unless it can be shown to be in *P* by a finite

number of uses of the base and induction rules.

---------------------------Respuesta----------------

1. *n*0 *∈ P* Por el caso base

2. Por la regla de inducción, *n*0 *∈ P → n*1 *∈ P* y así por *Modus Ponens*,

*n*1 *∈ P*.

3. Por la regla de inducción, *n*1 *∈ P → n*2 *∈ P* y así por *Modus Ponens*,

*n*2 *∈ P*.

4. Por la regla de inducción, *n*2 *∈ P → n*3 *∈ P* y así por *Modus Ponens*,

*n*3 *∈ P*.

**Exercise 24.** When is 0 in the set defined below?

**Definition 29.** Given a number *n*, the set *N* is defined as follows:

*• n ∈ N*

*• m ∈ N → m −* 2 *∈ N*

*•* Nothing is in *N* unless it can be shown to be in *N* by a finite number

of uses of the previous rules.

------------Respuesta-------------------------

--Si n es un numero positivo multiplo de 2 entonces sí, de otra forma no

**Exercise 25.** What set is defined by the following definition?

**Definition 30.** The set *S* is defined as follows:

*•* 1 *∈ S*

*• n ∈ S ∧ n mod* 2 = 0 *→ n* + 1 *∈ S*

*• n ∈ S ∧ n mod* 2 = 1 *→ n* + 2 *∈ S*

*•* Nothing else is in *S* unless it can be shown to be in *S* by a finite

number of uses of the previous rules.

-------------RESPUESTA--------------------------------------

--Caso base: 1 pertenece a S, 1 es impar.

--Caso inductivo: Asumiendo que cada número anterior ha sido un non,

-- n mod 2 será 1, entonces usando la regla 3: n + 2 pertenece a S

-- Desde que n es impar, n + 2 también es impar.

-- Por lo tanto el conjunto definido es el conjunto de números impares positivos.

**Exercise 26.** Prove that 4 is in the set defined as follows:

**Definition 31.** The set *S* is defined as follows:

1. 0 *∈ S*

2. *n ∈ S ∧ nmod*2 = 0 *→ n* + 2 *∈ S*

3. *n ∈ S ∧ nmod*2 = 1 *→ n* + 1 *∈ S*

4. Nothing is in *S* unless it can be shown to be in *S* by a finite number

of uses of the previous rules.

-------------RESPUESTA--------------------------------------

Por regla (1), 0 *∈ S*

Por regla (2) 0 *∈ S →* 2 *∈ S*  y así por *Modus Ponens* 2 *∈ S*.

Por regla (2) 2 *∈ S →* 4 *∈ S* so 4 *∈ S*. (2 mod 2=0 entonces 2+2=4, entonces 4 *∈ S).*

**Exercise 27.** Given the following definition, prove that the string ‘yyyy’ is in

*YYS*.

**Definition 32.** The set YYS of strings containing pairs of the letter ’y’

is defined as follows:

1. ""*∈ Y Y S*

2. *s ∈ Y Y S →* "*yy*"++*s ∈ Y Y S*

3. Nothing else is in YYS unless it can be shown to be in YYS by a

finite number of uses of rules (1) and (2).

------------------RESPUESTA----------------------

1. "" *∈ Y Y S*

2. "" *∈ Y Y S →* "*yy*" *∈ Y Y S* y así *Modus Ponens*, "*yy*" *∈ Y Y S*.

3. "*yy*" *∈ Y Y S →* "*yyyy*" *∈ Y Y S* y así por *Modus Ponens*, "*yyyy*" *∈ Y Y S (yyyy=yy++yy)*.

**Exercise 28.** Using data recursion, define the set of strings containing the letter ‘z’.

----------RESPUESTA---------

ex28 = "" : map ('z':) ex28

**Exercise 29.** Using induction, define the set of strings of spaces of length less than or equal to some positive integer *n*.

------------------RESPUESTA----------------------

1. ”” *∈ SS*

2. *ss ∈ SS ∧ length ss < n →*’ ’: *ss ∈ SS*

3. Nada pertenece a SS a menos que se demuestre que está en SS por un número finito de usos de las reglas (1) y (2).

**Exercise 30.** Using recursion, define the set of strings of spaces of length less than or equal to length *n*, where *n* is a positive integer.

------------RESPUESTA---------------------------

spaces :: Int -> [String]

spaces 0 = [""]

spaces n = take n (repeat ' ') : spaces (n - 1)

**Exercise 31.** We could have a set that consists of all the natural numbers except for 2; you can write this as *N − {*2*}*. Similarly, for every natural number *x*, there is a set that contains all the natural numbers except for *x*. Now, we could make a set *SSN* of all of these results. Write an

inductive definition of *SSN*.

------------RESPUESTA---------------------------

1. *N − {*0*} ∈ SSN*

2. (*N − {n}*) *∈ SSN →* (*N − {n* + 1*} ∈ SSN*

3. Nada está en *SSN* a menos que pueda ser demostrado que está en *SSN* por un número finito de usos de las reglas (1) and (2).

**Exercise 32.** Given the following definition, show that the set *I − {−*3*} ∈*

*SSI−*.

The set of sets of integers SSI, each of which is missing a distinct negative

integer, is defined inductively as follows:

1. *I − {−*1*} ∈ SSI−*

2. *I − {n} → I − {n −* 1*} ∈ SSI−*

3. Nothing else is in SSI- unless it can be shown to be in SSI- by a

finite number of uses of rules (1) and (2).

------------RESPUESTA---------------------------

1. I − {−1} ∈ SSI−

2. I − {−1} -> I − {−2} ∈ SSI−

3. I − {−2} -> I − {−3} ∈ SSI−

4. Por Modus Ponens, I − {−3} ∈ SSI−

**Exercise 33.** Given the following definition, prove that -7 is in ONI. The set

ONI of odd negative integers is defined as follows:

1. *−*1 *∈ ONI*

2. *n ∈ ONI → n −* 2 *∈ ONI*

3. Nothing is in ONI unless it can be shown to be in ONI by a finite

number of uses of the previous rules.

------------RESPUESTA---------------------------

1. −1 ∈ ONI

2. −1 ∈ ONI -> −3 ∈ ONI entonces −3 ∈ ONI

3. −3 ∈ ONI -> −5 ∈ ONI entonces −5 ∈ ON.

4. −5 ∈ ONI -> −7 ∈ ONI entonces −7 ∈ ONI

**Exercise 34.** Using data recursion, define the set ni of negative integers.

------------RESPUESTA---------------------------

ni = -1 : map decrement ni

where decrement x = x - 1

-- Al ejecutar \*Main>ni

-- [-1,-2,-3,-4,-5,-6,-7,-8,-9,-10, -11.......]

**Exercise 35.** If you print the elements of

[(a,b) | a <- [0..], b <- [0..]]

will you ever see the element (1,2)?

-------------RESPUESTA---------------

--No se visualiza el elemento (1,2), porque el 0 debe ser emparejado con cada elemento en [0..], pero este emparejamiento no tiene fin, por lo que no se visualizará el elemento (1,2).

**Exercise 36.** What set is given by the following definition?

**Definition 33.** The set *S* is defined as follows:

1. 1 *∈ S*

2. *n ∈ S → n − n ∈ S*

3. Nothing is in *S* unless it can be shown to be in *S* by a finite number

of uses of the previous rules.

---------------RESPUESTA---------------------

-- El conjunto {0,1}

-- 1  *∈*  S -> 1 - 1 = 0  *∈*  S

-- 0 - 0 = 0  *∈* S