Actividad extra de parcial # 1

Inciso # 1: Demuestre que para todo natural de peano "n" se cumple la siguiente propiedad:

$$\textbf{Succ 0 + n = Succ n}$$

Teniendo las siguientes propiedades de la suma:

1. $n+0 = 0$
2. $0+m = m$
3. $n + \text{Succ } a = \text{Succ } (n+a)$
4. $a + \text{Succ } b = \text{Succ } a + b$

Podemos hacer lo siguiente:

1. $\text{Succ } 0 + n$
2. $n + \text{Succ } 0$ (Por medio de la propiedad conmutativa)
3. $\text{Succ } (n + 0)$
4. $\text{Succ } n$ (De esta manera queda demostrado)

Inciso # 2: Provea una definición inductiva para la propiedad "mayor que" (>) tal que:

$$a > b \begin{cases} \text{Succ } 0 & \text{si } a \text{ es mayor que } b \\ 0 & \text{de lo contrario} \end{cases}$$

En otras palabras, la propiedad "mayor que" es equivalente a Succ 0 si el primer valor es mayor que el segundo o 0 de lo contrario. Puede utilizar el operador ">" en su definición de la misma manera que se utiliza "+" en la definición de suma.

1. $0 > 0 = 0$
2. $a > 0 = 1$
3. $0 > m = 0$
4. $\text{Succ } a > \text{Succ } b = a > b$

Inciso # 3: Provea una definición de las propiedades "esPar" e "esImpar" tal que:

$$esImpar \begin{cases} Succ\ 0 & Si\ n\ es\ impar \\ 0 & de\ lo\ contrario \end{cases}$$

$$esPar\ n \begin{cases} Succ\ 0 & si\ n\ es\ un\ numero\ par \\ 0 & de\ lo\ contrario \end{cases}$$

esPar 0 = 1
esPar 1 = 0
esPar (Succ a) = esImpar a

esImpar 0 = 0
esImpar 1 = 1
esImpar (Succ a) = esPar a

Serie #4: Utilice el lenguaje de programación Haskell para definir la propiedad "predecesor". Esta propiedad debe aceptar un numero de peano y producir el predecesor de este. En el caso de cero, utilizar cero como su predecesor.

```
{-# LANGUAGE NoImplicitPrelude #-}

module Main where
import Prelude (Show, undefined)

data Natural = Cero | Succ Natural deriving Show

-- Suma
Cero + m = m
n + Cero = n
n + (Succ a) = Succ (n + a)

-- Regla multiplicación
n * Cero = Cero
n * Succ Cero = n
n * Succ a = n + (n * a)


--predecesor de un número
predecesor Cero = Cero
predecesor (Succ Cero) = Cero
predecesor (Succ a) = a

main = undefined
```