Introducción a la programación

Práctica 4: Recursión sobre números enteros

Ejercicio 13

Especificar e implementar la siguiente función:

$$f(n, m) = \sum_{i=1}^{n} \sum_{j=1}^{m} i^j$$

Ejercicio 13

Especificar e implementar la siguiente función:

$$f(n,m) = \sum_{i=1}^{n} \sum_{j=1}^{m} i^j$$

Ejemplos:
f(1,1)=1
f(1,2)=1+1=2
f(1,3)=1+1+1=3
f(2,1)=1+2=3
f(2,2)=1+1+2+4=8

La especificación, esta ok?

```
problema dobleSumaDePotencias (n,m ℤ) : ℤ {
  requiere: { True }
  asegura: { resultado = \sum_{i=1}^{n} \sum_{j=1}^{m} i^j }
}
```

La especificación, ojo con los requiere:

problema dobleSumaDePotencias (n:,m $\mathbb{Z}$) : $\mathbb{Z}$ {

  requiere: $\{ n >= 1 \}$

  requiere: $\{ m >= 1 \}$

  asegura: $\{ resultado = \sum_{i=1}^{n} \sum_{j=1}^{m} i^j \}$

}

Ejercicio 13

¿Como pensamos esta doble sumatoria usando recursión?
La sumatoria se puede pensar separando cada pieza:
$f(n,m) = \sum_{i=1}^{n} \sum_{j=1}^{m} i^j =$

Podemos sumar el caso i = n separado:
$= \sum_{i=1}^{n-1} \sum_{j=1}^{m} i^j + \sum_{j=1}^{m} n^j$

Podemos entonces definir una función que sume todos los elementos desde 1 hasta m, de n^j.
¿Cuál será el caso base de sumar todas las potencias de n desde 1 hasta m? ¿Cuál el caso recursivo de esta función auxiliar que nos servirá para resolver la función más grande?

Posible solución Ej. 13

Sumamos todas las potencias de n, desde 1 hasta m:

```
sumaPotenciasDe :: Integer -> Integer -> Integer
sumaPotenciasDe n 1 = n
sumaPotenciasDe n m = n^m + sumaPotenciasDe n (m-1)
```

Posible solución Ej. 13

Sumamos todas las sumas de potencias de 1..n, desde 1 hasta m:

```
sumaDoblePotencias :: Int -> Int -> Int
sumaDoblePotencias 1 m = sumaPotenciasDe 1 m
sumaDoblePotencias n m = sumaPotenciasDe n m
                         + sumaDoblePotencias (n-1) m
```

Ejercicio 16

- Implementar `menorDivisor :: Integer -> Integer` que calcule el menor divisor (mayor que 1) de un natural n pasado como parámetro.

Ej16. Una posible solución

```haskell
menorDivisor :: Integer -> Integer
menorDivisor 1 = 1
menorDivisor n = menorDivisorHasta n 2

menorDivisorHasta :: Integer -> Integer -> Integer
menorDivisorHasta n i
     | (mod n i == 0) = i
     | otherwise = menorDivisorHasta n (i+1)
```

Ejercicio 19

Implementar la función `esSumaInicialDePrimos :: Int -> Bool` según la siguiente especificación:

`problema` `esSumaInicialDePrimos` (n: $\mathbb{Z}$) : $\mathbb{B}$ $\{$
  `requiere:` $\{\ n \geq 0\ \}$
  `asegura:` $\{\ resultado = true \leftrightarrow n$ es igual a la suma de los m
    primeros números primos, para algún m.$\}$
$\}$

Para resolver el 19 debemos pensar la recursión

Para saber si n es la suma inicial de primos, debemos verificar si n es la suma inicial de alguno de los primeros k primos.
Para calcular la sumatoria de los primeros K primos, debemos saber cual es el `nEsimoPrimo` (ejercicio 16d). Y a su vez para resolver el anterior vamos a necesitar el ejercicio 16b `esPrimo`.
Volviendo a nuestra función original, definimos otra función **recursiva** para verificar si la sumatoria de los primeros K primos desde el primer primo es efectivamente el n que estamos buscando.
Esta recursión será desde 1 hasta que la suma de los primos supere a n -en ese caso ya sabemos que no hay sumatoria de primos que sea igual a n.

Ej19. Una posible solución

```
esSumaInicialDePrimos :: Int -> Bool
esSumaInicialDePrimos n = esSumaDePrimerosKPrimos 1 n

esSumaDePrimerosKPrimos :: Int -> Int -> Bool
esSumaDePrimerosKPrimos k n
    | (sumaKprimos k) == n = True
    | (sumaKprimos k) > n = False
    | otherwise = esSumaDePrimerosKPrimos (k+1) n
```