# Introducción a la Programación

# Algoritmos y Estructuras de Datos I

Primer cuatrimestre de 2025

Departamento de Computación - FCEyN - UBA

Práctica 4: Recursión sobre números enteros - Parte 2

# Ejercicio 14

Especificar e implementar una función `sumaPotencias :: Integer -> Integer -> Integer -> Integer` que dados tres naturales $q, n, m$ sume todas las potencias de la forma $q^{a+b}$ con $1 \leq a \leq n$ y $1 \leq b \leq m$.

# Ejercicio 14

Especificar e implementar una función `sumaPotencias :: Integer -> Integer -> Integer -> Integer` que dados tres naturales $q, n, m$ sume todas las potencias de la forma $q^{a+b}$ con $1 \leq a \leq n$ y $1 \leq b \leq m$.
Ejemplos:
$sumaPotencias(1, 1, 1) =$

# Ejercicio 14

Especificar e implementar una función `sumaPotencias :: Integer -> Integer -> Integer -> Integer` que dados tres naturales $q, n, m$ sume todas las potencias de la forma $q^{a+b}$ con $1 \leq a \leq n$ y $1 \leq b \leq m$.

Ejemplos:

$sumaPotencias(1, 1, 1) = 1^{1+1} = 1$

$sumaPotencias(2, 1, 2) =$

# Ejercicio 14

Especificar e implementar una función `sumaPotencias :: Integer -> Integer -> Integer -> Integer` que dados tres naturales $q, n, m$ sume todas las potencias de la forma $q^{a+b}$ con $1 \leq a \leq n$ y $1 \leq b \leq m$.

Ejemplos:

$sumaPotencias(1, 1, 1) = 1^{1+1} = 1$

$sumaPotencias(2, 1, 2) = 2^{1+1} + 2^{1+2} = 4 + 8 = 12$

$sumaPotencias(3, 3, 1) =$

# Ejercicio 14

Especificar e implementar una función `sumaPotencias :: Integer -> Integer -> Integer -> Integer` que dados tres naturales $q, n, m$ sume todas las potencias de la forma $q^{a+b}$ con $1 \leq a \leq n$ y $1 \leq b \leq m$.

Ejemplos:

$sumaPotencias(1,1,1) = 1^{1+1} = 1$

$sumaPotencias(2,1,2) = 2^{1+1} + 2^{1+2} = 4 + 8 = 12$

$sumaPotencias(3,3,1) = 3^{1+1} + 3^{2+1} + 3^{3+1} = 9 + 27 + 81 = 117$

$sumaPotencias(2,2,2) =$

# Ejercicio 14

Especificar e implementar una función `sumaPotencias :: Integer -> Integer -> Integer -> Integer` que dados tres naturales $q, n, m$ sume todas las potencias de la forma $q^{a+b}$ con $1 \leq a \leq n$ y $1 \leq b \leq m$.

Ejemplos:

$sumaPotencias(1, 1, 1) = 1^{1+1} = 1$

$sumaPotencias(2, 1, 2) = 2^{1+1} + 2^{1+2} = 4 + 8 = 12$

$sumaPotencias(3, 3, 1) = 3^{1+1} + 3^{2+1} + 3^{3+1} = 9 + 27 + 81 = 117$

$sumaPotencias(2, 2, 2) = 2^{1+1} + 2^{1+2} + 2^{2+1} + 2^{2+2} = 4 + 8 + 8 + 16 = 36$

# Ejercicio 14 - Especificación

¿Hay que tener en cuenta alguna precondición?

# Ejercicio 14 - Especificación

¿Hay que tener en cuenta alguna precondición? Notar que el enunciado dice tres $naturales$, luego podemos pensar la precondición como $n > 0 \wedge m > 0 \wedge q > 0$

# Ejercicio 14

**¿Como pensamos este problema usando recursión?**
Es similiar al ejercicio 13 que vimos en la clase teórica, podemos pensarlo como $sumaPotencias(q, n, m) = \sum_{i=1}^{n} \sum_{j=1}^{m} q^{i+j}$

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

Notemos que en el último término, $n$ es una constante. Luego para resolverlo alcanza con hacer recursión sobre $m$

# Ejercicio 14 - Posible solución

## Posible solución

Se mostrará en clase

# Ejercicio 16

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

# Ejercicio 16 - Posible solución

### Posible solución

Se mostrará en clase

# 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 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 cuál es el `nEsimoPrimo` (ejercicio 16d). Y a su vez para resolver el anterior vamos a necesitar el ejercicio 16b `esPrimo`.

# 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 cuál 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, ya que 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 = esSumaDePrimerosKPrimo
```

Falta definir `sumaKprimos`, que probablemente use varias funciones auxiliares!

# Ejercicio 16 - Posible solución

## Posible solución

Se mostrará en clase