Nombre y apellido:	L.U. o D.N.I.:	
Carrera:	Número de órden:	Cant. de hojas:

Departamento de Computación – FCEyN – UBA

Taller de Álgebra I - Parcial

SEGUNDO CUATRIMESTRE 2017 – TURNO MIÉRCOLES AM

18 de octubre de 2017

Aclaraciones

- El parcial se aprueba con tres ejercicios bien resueltos.
- Programe todas las funciones en lenguaje Haskell. El código debe ser autocontenido. Si utiliza funciones que no existen en Haskell, debe programarlas.
- Incluya la signatura de todas las funciones que escriba.
- No está permitido: alterar los tipos de datos presentados en el enunciado – utilizar técnicas no vistas en clase para resolver los ejercicios.

Ejercicio 1

Implementar una función `distanciaManhattan:: (Float, Float, Float) -> (Float, Float, Float) -> Float` que dados dos vectores en $\mathbb{R}^3$ calcula la distancia Manhattan entre ambos. La distancia Manhattan entre dos vectores se define como $d(\mathbf{p}, \mathbf{q}) = \sum_{i=1}^{3} |p_i - q_i|$, donde p_i, q_i son las i-ésimas coordenadas del vector $\mathbf{p} = (p_1, p_2, p_3)$ y $\mathbf{q} = (q_1, q_2, q_3)$ respectivamente.a

Por ejemplo:

```
distanciaManhattan (2, 3, 4) (7, 3, 8) ⇝ 9
distanciaManhattan ((-1), 0, (-8.5)) (3.3, 4, (-4)) ⇝ 12.8
```

Ejercicio 2

Implementar la funcion `esDefectivo :: Integer -> Bool` que dado un $n \in \mathbb{N}_{>0}$ determine si es defectivo, esto quiere decir, que cumple que la suma de sus divisores propios positivos es menor que el propio número.

Por ejemplo:

```
esDefectivo 16 ⇝ True
esDefectivo 12 ⇝ False
```

Ya que la suma de los divisores propios de 16 es $1 + 2 + 4 + 8 = 15$, que es menor que 16; y la suma de los divisores propios de 12 es $1 + 2 + 3 + 4 + 6 = 16$, que es mayor que 12.

Ejercicio 3

Implementar una función `dobleSumatoria :: Integer -> Integer -> Integer` que para $n, m \in \mathbb{N}_{>0}$ calcule el resultado de la sumatoria doble $\sum_{i=1}^{2n} \sum_{j=n}^{m} (i + j)^5$.

Ejercicio 4

Asumiendo que disponemos de la función `esPrimo :: Integer -> Bool` que decide si un entero es primo o no, implementar una función `soloPrimos :: [Integer] -> [Integer]` que dada una lista de números enteros positivos, devuelve la lista que sólo contiene aquellos números de la lista original que son primos.

Por ejemplo:

```
soloPrimos [2,4,65,33,23,2,1,3] ⇝ [2,23,2,3]
```

Ejercicio 5

Programe la función `comprimir :: [Integer] -> [(Integer, Integer)]` que dada una lista de números enteros devuelva una lista que contenga una tupla (número, cantidad de apariciones) por cada ráfaga de números iguales adyacentes.

Por ejemplo:

```
comprimir [1,1,7,7,4,4,1,4,4,4,3,3,3] ⇝ [(1,2),(7,2),(4,2),(1,1),(4,3),(3,3)]
```

Sugerencia: Empiece reemplazando cada número n por una tupla $(n, 1)$.