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 PM

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 la función `todoMenor :: (Float, Float, Float) -> (Float, Float, Float) -> Bool` que dados dos vectores $x, y \in \mathbb{R}^3$ decida si todos los elementos de x son menores a todos los de y.

Por ejemplo:

```
todoMenor (3,-1,2) (5,10,0) ⇝ False
todoMenor (4,-1,7) (5,21,5) ⇝ True
todoMenor (2,1,31) (40,61,15) ⇝ False
```

Ejercicio 2

Implementar una función `sumaFibonacci :: Integer -> Integer` que para cada $n \geq 1$ calcule $\sum_{j=0}^{n} f_n$, donde f_n es n–ésimo término de la sucesión de Fibonacci.

Por ejemplo:

```
sumaFibonacci 2 ⇝ 1+1+2 ⇝ 4
sumaFibonacci 4 ⇝ 1+1+2+3+5 ⇝ 12
```

Ejercicio 3

Asumiendo que disponemos de la función `esPrimo :: Integer -> Bool` que decide si un entero es primo o no, implementar una función `esBSuave :: Integer -> Float -> Bool` que, dados un entero positivo n y un real positivo B, decida si todos los primos que dividen a n son menores a B.

Por ejemplo:

```
esBSuave 1 90 ⇝ True        (pues no hay primos que dividan a 1)
esBSuave 81 4 ⇝ True
esBSuave 21 6 ⇝ False
```

Ejercicio 4

Programar la función `congruenciasMod3:: [Integer] -> (Integer, Integer, Integer)` que dada una lista (finita) de números enteros L devuelve una terna de enteros cuyo i-ésimo elemento, entendiendo al primero como el 0-ésimo, es la cantidad de elementos de L que son congruentes a i módulo 3.

Por ejemplo:

```
congruenciasMod3 [-9,-1,5,0,8,2,1,3] ⇝ (3,1,4)
```

Ejercicio 5

Implementar una función `esSumaMod7DeDos :: Integer -> [Integer] -> Bool` que, dados un entero k y una lista (finita) de números enteros L, decida si k es igual, *módulo 7*, a la suma de dos elementos de la lista L.

Por ejemplo:

```
esSumaMod7DeDos 1 [1,-2,3,5,2] ⇝ True     (pues 1 ≡ 3 + 5  mód 7)
esSumaMod7DeDos 2 [1,2,4] ⇝ False         (no admitimos sumar al 1 consigo mismo)
esSumaMod7DeDos 5 [6,-3,6,2] ⇝ True       (pues 5 ≡ 6 + 6  mód 7; sí admitimos sumar dos apariciones de
un mismo número en lugares distintos de la lista)
```