Autómatas y Lenguajes Formales 2019-2
Facultad de Ciencias UNAM
Ejercicio Práctico 0

Favio E. Miranda Perea Javier Enríquez Mendoza

18 de Febrero del 2019
Fecha de entrega: 25 de Febrero del 2019

El ejercicio debe resolverse en un archivo `ejp00aylf192.hs` y en el lenguaje de programación Haskell.

1. Definir la función `sFibonacci` que recibe un entero n y regresa una lista con los primeros n + 1 elementos de la sucesión de Fibonacci.

```
sFibonacci :: Int -> [Int]
```

```
> sFibonacci 12
[0,1,1,2,3,5,8,13,21,34,55,89,144]
> sFibonacci 15
[0,1,1,2,3,5,8,13,21,34,55,89,144,233,377,610]
```

2. Definir la función `quitaElemento` que recibe una lista de elementos comparables y un elemento de la lista y regresa una nueva lista sin ninguna aparición de este elemento.

```
quitaElemento :: (Eq a) => [a] -> a -> [a]
```

```
> quitaElemento [1,2,3,4,5,1,2,3,4,5,1,1,1] 1
[2,3,4,5,2,3,4,5]
> quitaElemento ['a','a','a','a','a','a'] 'a'
[]
```

3. Definir usando listas por comprensión la función `divisoresPropios` que recibe un entero y regresa una lista con los divisores propios de éste. Los divisores propios de un número n son todos los enteros distintos de n que son divisores de n.

```
divisoresPropios :: Int -> [Int]
```

```
> divisoresPropios 25
[1,5]
> divisoresPropios 1725
[1,3,5,15,23,25,69,75,115,345,575]
```

4. Un *número perfecto* es un número natural que es igual a la suma de sus divisores propios positivos. Por ejemplo, 6 es un número perfecto porque sus divisores propios son 1, 2 y 3; y $6 = 1 + 2 + 3$. Definir la función `esPerfecto` que reciba un entero positivo y diga si es perfecto o no.

```
esPerfecto :: Int -> Bool
```

```
> esPerfecto 496
True
> esPerfecto 2500
False
```

5. Dos *números amigos* son dos números enteros positivos a y b tales que la suma de los divisores propios de uno es igual al otro número y viceversa, esto es, si `sumaDP` es función que suma los divisores propios de un número, entonces se cumple que `sumaDP(a) = b` y `sumaDP(b) = a`. Por ejemplo 220 y 284 son amigos pues `sumaDP(220) = 284` y `sumaDP(284) = 220`. Definir la función `sonAmigos` que recibe dos enteros positivos y determina si son amigos.

```
sonAmigos :: Int -> Int -> Bool
```

```
> sonAmigos 1184 1210
True
> sonAmigos 114 110
False
```

6. Definir la función `supersuma` que recibe un entero y regresa la suma de sus dígitos hasta que quede un número de un solo dígito.

```
supersuma :: Int -> Int
```

```
> supersuma 28
1
> supersuma 3765
3
```

7. Definir las funciones `reversar` y `reversal` que obtienen la reversa de una lista usando las funciones de orden superior `foldr` y `foldl` respectivamente. Para este ejercicio no se pueden definir funciones auxiliares, es necesario el uso de `lambdas`.

```
reversar :: [a] -> [a]
reversal :: [a] -> [a]
```

```
> reversar [1,2,3,4,5]
[5,4,3,2,1]
> reversal [3,7,6,5]
[5,6,7,3]
```

Entrega.

- Se entrega antes de las 23:59 horas del día fijado como fecha de entrega.
- El ejercicio debe ser entregado de forma individual.
- Debe incluirse un archivo README.txt con el nombre completo del alumno, así como comentarios, opiniones o ideas sobre el ejercicio.
- Guardar los archivos requeridos en un directorio que tenga como nombre el número de cuenta del alumno. Se entregará éste directorio comprimido.
- Entregarse al correo del ayudante con el asunto [AyLF192-EP00].
- **Cualquier copia o plagio será calificado automáticamente con cero.**