

**Universidad Nacional Autónoma de México**  
**Facultad de Ciencias**

**Estructuras Discretas**

**Práctica 2**

**Javier Enríquez Mendoza**

31 de Agosto de 2018

**Fecha de entrega:** 14 de Septiembre de 2018

## Objetivos

- Familiarizar al alumno con el lenguaje de programación Haskell.
- Definir funciones en Haskell.
- Reforzar los conocimientos adquiridos en el laboratorio.

## Instrucciones generales

La práctica debe resolverse en un archivo `Practica2.hs` y las firmas de las funciones deben ser idénticas a las que se muestran en cada ejercicio. Cada función debe estar debidamente comentada con la especificación de ésta.

Se tomará en cuenta la legibilidad y el estilo del código. **No** pueden usarse funciones sobre listas predefinidas del lenguaje, en caso de ser necesarias definir una versión propia de ésta.

## Ejercicios

### 1 Listas (1.5 ptos)

Representar los siguientes conjuntos mediante listas por comprensión y rangos.

**1.1** Los números naturales.

**1.2** Múltiplos de 10.

**1.3** Potencias de 2.

**1.4** Los números pares.

**1.5** Los años desde el año de nacimiento del alumno hasta el actual.

## 2 Funciones

**Ejercicio 2.1 (1 pt.)** Definir la función `fibonacci` que recibe un número y regresa el fibonacci de éste.

```
fibonacci :: Int -> Int
```

```
> fibonacci 12
144
> fibonacci 25
75025
```

**Ejercicio 2.2 (1 pt.)** Definir la función `elemento` que recibe una lista de elementos comparables y un posible elemento de la lista y verifica si el elemento pertenece a la lista. **No** se puede utilizar **ninguna** función de listas predefinida en Haskell.

```
elemento :: (Eq a) => [a] -> a -> Bool
```

```
> elemento [1..] 15
True
> elemento ['Hola', 'Hello', 'Hallo', 'Bonjour'] 'Ciao'
False
```

**Ejercicio 2.3 (1 pt.)** Definir la función `sumaLista` que recibe una lista de números y regresa la suma de todos éstos. **No** se puede utilizar **ninguna** función de listas predefinida en Haskell.

```
sumaLista :: (Num a) => [a] -> a
```

```
> sumaLista [1,2,3,4,5,6,7,8,9]
45
> sumaLista [2.0,3.0,5.0,7.0,11.0]
28.0
```

**Ejercicio 2.4 (1 pt.)** Definir la función `meses` que recibe una lista de enteros del 1 al 12 y regresa una lista de cadenas con el mes correspondiente a cada uno de los enteros de la lista original.

**Hint:** Utilizar la función `mes` de la práctica anterior.

```
meses :: [Int] -> [String]
```

```
> meses [1,3,5,7]
['Enero','Marzo','Mayo','Julio']
> meses [10,11,12]
['Octubre','Noviembre','Diciembre']
```

**Ejercicio 2.5 (1 pt.)** Definir 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]
```

**Ejercicio 2.6 (1 pt.)** 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
```

**Ejercicio 2.7 (1 pt.)** 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
```

**Ejercicio 2.8 (1 pt.)** Definir la función `supersuma` que recibe un entero y regresa la suma de sus dígitos.

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

```
> supersuma 28
10
> supersuma 3765
21
```

**Ejercicio 2.9 (1 pt.)** Los números del 0 al 9 en japones se nombran de la siguiente manera:

0	→	<i>rei</i>
1	→	<i>ichi</i>
2	→	<i>ni</i>
3	→	<i>san</i>
4	→	<i>yon</i>
5	→	<i>go</i>
6	→	<i>roku</i>
7	→	<i>nana</i>
8	→	<i>haci</i>
9	→	<i>kyu</i>

El número 10 se nombra como *ju* y a partir de éste, pueden construirse los números del 11 al 99. Basta con indicar cuantas veces se suma el diez y cuantas unidades tiene. Por ejemplo:

- 20 se nombra *ni ju* pues se suma dos (*ni*) veces diez (*ju*).
- 37 se nombre *san ju nana* pues se suma tres (*san*) veces diez (*ju*) y se tienen siete (*nana*) unidades.
- 83 se nombra *haci ju san* pues se suma ocho (*haci*) veces diez (*ju*) y se tienen tres (*san*) unidades.

Definir la función `japones` que recibe un número entre 0 y 99 y regresa una cadena con su representación japonesa.

```
japones :: Int -> String
```

```
> japones 68  
"roku ju haci"  
> japones 37  
"san ju nana"
```

## Entrega

- La entrega se realiza mediante correo electrónico a la dirección del ayudante de laboratorio (javierem\_94@ciencias.unam.mx).
- La practica deberá ser entregada de forma individual.
- Se debe entregar un directorio `numeroCuenta_P02`, dónde `numeroCuenta` es el número de cuenta del alumno. Dentro del directorio se debe incluir:
  - \* Un archivo `readme.txt` con el nombre y número de cuenta del alumno, comentarios, opiniones, críticas o ideas sobre la práctica.
  - \* Los archivos requeridos en la práctica. Debe enviarse código lo más limpio posible.
- El único archivo requerido es `Practica2.hs`.
- El directorio se deberá comprimir en un archivo con nombre `numeroCuenta_P02.tar.gz`, dónde `numeroCuenta` es el número de cuenta del alumno.
- El asunto del correo debe ser `[ED-20191-P02]`.
- Se recibirá la práctica hasta las 23:59:59 horas del día fijado como fecha de entrega, cualquier práctica recibida después no será tomada en cuenta.
- **Cualquier práctica total o parcialmente plagiada, será calificada automáticamente con cero y no se aceptarán más prácticas durante el semestre.**