Lenguajes de programación 2016-2
Práctica 0

Noé Salomón Hernández Sánchez
Albert M. Orozco Camacho
C. Moisés Vázquez Reyes

Facultad de Ciencias UNAM

Algunos de los ejercicios se resolverán en la sesión de laboratorio.

1. Números naturales

Considera el tipo:

```
data N = Zero | Suc N deriving Show
```

el cual representa a los números naturales.

1. `suma::N->N->N`

   Realiza la suma de dos números naturales.

2. `prod::N->N->N`

   Realiza el producto de dos números naturales.

3. `pot::N->N->N`

   Calcula la potencia n^m, donde n es el primer argumento y m el segundo.

4. `menor::N->N->Bool`

   Compara dos números naturales; devuelve *True* cuando el primer argumento es menor que el segundo, *False* en otro caso.

5. `igual::N->N->Bool`

   Compara dos números naturales; devuelve *True* cuando ambos números son iguales, *False* en otro caso.

2. Números DNat

Un número DNat es:

- 0 es un DNat.
- Si n es un DNat, $D(n)$ es un DNat.
- Si n es un DNat, $U(n)$ es un DNat.

donde $D(n)$ denota el *doble* de n y $U(n)$ el *sucesor del doble* de n.

La respectiva definición de DNat en Haskell es la siguiente:

```
data DNat = Cero | D DNat | U DNat deriving Show
```

Usando la definición de DNat, así se construyen los primeros seis números naturales:

- $0 := 0$
- $1 := U\ 0$
- $2 := D(U\ 0)$
- $3 := U(U\ 0)$
- $4 := D(D(U\ 0))$
- $5 := U(D(U\ 0))$

3. Ejercicios:

1. `simplDN::DNat->DNat`

   Simplifica la representación de un DNat; el cero tiene una infinidad de representaciones visto como DNAT, por ejemplo: $D\ Cero$, $D\ (D\ Cero)$, ...

   Ejemplos:

   - ```
     *Main>simplDN (U (D (D (D Cero))))
     U Cero
     ```
   - ```
     *Main>simplDN (D (U (D (D Cero))))
     D (U Cero)
     ```

2. `sucDN::DNat->DNat`

   Calcula el sucesor de un número DNat.

   Ejemplos:

   - `*Main>sucDN (D (U (D Cero))`
     `U (U Cero)`
   - `*Main>sucDN (U (U Cero))`
     `D (D (U Cero))`

3. `predDN::DNat->DNat`

   Calcula el predecesor de un número DNat.

   Ejemplos:

   - `*Main>predDN (D (U Cero))`
     `U Cero`
   - `*Main>predDN (U (U (D Cero)))`
     `D (U Cero)`

4. `dNToZ::DNat->Int`

   Nos da la representación de un número DNat en los números enteros.

   Ejemplos:

   - `*Main>dNToZ (U (U (D (D (U (U (D (D (D (D (D (U Cero))))))))))))`
     `2099`
   - `*Main>dNToZ (U (D (D (U (D (D (U (D (D (D (U (U Cero))))))))))))`
     `3145`

5. `sumaDN::DNat->DNat->DNat`

   Función que calcula la suma dos números DNat. No se vale convertir los argumentos a números enteros, sumarlos y convertir el resultado a un DNat.

   Ejemplos:

   - `*Main>sumaDN (D (U (D Cero))) (D (U Cero))`
     `D (D (U Cero))`
   - `*Main>sumaDN (D (U Cero)) (U (U (D Cero)))`
     `U (D (U Cero))`

6. `prodDN::DNat->DNat->DNat`

   Función que calcula el producto dos números DNat. No se vale convertir los argumentos a números enteros, hacer el producto y convertir el resultado a un DNat. Ejemplos:

   - `*Main>prodDN (D (U (D Cero))) (D (U Cero))`
     `D (D (U Cero))`
   - `*Main>prodDN (D (U Cero)) (U (U (D Cero)))`
     `D (U (U Cero))`

7. `zToDNat::Int->DNat`

   Transforma un entero positivo a su representación en DNat.

   Ejemplos:

   - `*Main>zToDNat 616`
     `D (D (D (U (D (U (U (D (D (U Cero))))))))`
   - `*Main>zToDNat 1610`
     `D (U (D (U (D (D (U (D (D (U (U Cero))))))))))`

4. Listas

1. `toSet::Eq a=>[a]->[a]`

   Toma una lista y la hace conjunto, es decir, elimina los elementos repetidos de una lista

   Ejemplos:

   - `*Main>toSet [1,1,2,4,2,4,3]`
     $[1, 2, 4, 3]$
   - `*Main>toSet [3,4,2,1]`
     $[3, 4, 2, 1]$

2. `cuantas::Eq a=>a->[a]->Int`

   Nos dice cuántas veces aparece x en ℓ

   Ejemplos:

   - `*Main>cuantas 1 [1,2,3,1,4]`
     2

- `*Main>cuantas 7 [3,4,2,1]`
  0

3. `frec::Eq a=>[a]->[(a,Int)]`

   Dada una lista ℓ, nos devuelve la siguiente lista:

   $$\{(x, y)|x \in \ell,\ y = \text{el número de veces que aparece } x \text{ en } \ell\}$$

   Ejemplos:

   - `*Main>frec [1,2,3,1,4]`
     $[(2, 1), (3, 1), (1, 2), (4, 1)]$
   - `*Main>frec [11,8,3,1,4,4]`
     $[(11, 1), (8, 1), (3, 1), (1, 1), (4, 2)]$

4. `unaVez::Eq a=>[a]->[a]`

   Dada una lista ℓ, nos devuelve a los elementos que aparecen sólo una vez en ℓ

   Ejemplos:

   - `*Main>unaVez [1,2,3,1,4]`
     $[2, 3, 4]$
   - `*Main>unaVez [11,8,3,1,4,4]`
     $[11, 8, 3, 1]$

5. Retos

1. `compress1::String->String`

   La entrada es una cadena que contiene palabras separadas por espacios, debes tomar la primer letra de cada palabra y juntarlas en una sola cadena.

   Es decir, para la entrada:

   Remember when I moved in you; the holy dark was moving too, and every breath we drew was Hallelujah

   debes devolver:

   RwImiythdwmtaebwdwH

2. `compress2::String->String`

   La entrada es una cadena que contiene palabras separadas por espacios, cada palabra tiene un número al inicio. De cada palabra debes obtener el caracter que esté en la posición que el número al inicio indique y debes devolverlos en una sola cadena. Si el número excede la longitud de la palabra debes devolver un espacio en blanco.

   Es decir, para la entrada:

   1Las 0cosas 22más 3importantes 2son 4siempre 543las 42mas 8difíciles 1de 3contar. 3Creo 8que 5eso 4es 5precisamente 193lo 3peor, 4que 8el 4secreto 1lo 23siga 2siendo, 44no 5por 1falta 0de 91un 4narrador, 3sino 2por 4falta 2de 3un 5555oyente 8comprensivo

   debes devolver:

   ac onp seto s r eo e ad aora i

   Recomiéndale una buena canción a tu ayudante de laboratorio.