**TALLER HASKELL**

1. Debe encontrar el orden del entero 8 módulo 13, que se define como:

$$ord(8) = \min(t : 8^t \equiv 1 \; mod \; 13 \;)$$

es decir, debe encontrar el entero $t$ más pequeño que satisfaga la congruencia

$$8^t \equiv 1 \; mod \; 13$$

teniendo en cuenta que $t \in \{ \; 1, 2, 3, \dots, 13 \; \}$

2. ¿Cuál es el error del siguiente trozo de código en Haskell?

```
Prelude> let a = [ if x < 10 then "BAM! " | x <- [ 5 .. 15 ] ]

<interactive>:36:34: error: parse error on input ‘|’
```

3. Implemente una función la cual tiene como argumento de entrada un valor numérico ***n***, dicha función calculará la siguiente recurrencia:

$$0 * 1 + 2 * 3 + 4 * 5 + 6 * 7 + \cdots + (n-1) * n$$

Su función debe retornar una lista que contenga la solución de la recurrencia hasta cada uno de los parámetros, ordenado desde el último valor hasta el primero.

**Ejemplos**

$$n = 5 \rightarrow [ \; 50, 10, 6, 2, 0, 0 \; ]$$

$$n = 10 \rightarrow [ \; 3610, 3600, 400, 392, 56, 50, 10, 6, 2, 0, 0 \; ]$$

¿Cuál es el número en la posición 0 de la lista para $n = 65$ ?

**Sugerencia:** No es necesario imprimir la lista, acceda a la posición con el operador **!!**

```
Prelude> let list = [ 1, 2, 3 ]
Prelude> list !! 2
3
```

4. Dada una lista de números integrales su tarea consiste en crear una función que imite el resultado de la función ***map***, n veces.

   La función tiene el siguiente esquema:

```
mapNTimes :: ( Integral a ) => ( a -> a ) -> [ a ] -> a -> [ a ]
```

   Recibirá una función, una lista de números integrales y un número que determinará cuántas veces se debe aplicar la función a cada elemento de la lista.

   Note que la función a aplicar recibe parámetros integrales y retorna datos del mismo tipo.

   **Ejemplos:**

```
Prelude> mapNTimes (+3) [ 1, 2, 3 ] 3
[10,11,12]
```

   La función ha retornado 10, 11, 12 puesto que se adicionó 3 veces 3 a cada número de la lista.

```
Prelude> mapNTimes succ [ 1, 2, 3 ] 3
[4,5,6]
```

   La función ha retornado 4, 5, 6 puesto que se consultó el tercer sucesor de cada elemento de la lista.

   **Nota:** No se permite el uso de la función ***map***.

   **Sugerencia:** Considere usar una función aparte que aplique una función $f$ sobre un parámetro *x,* ***n*** **veces**.

   Para la siguiente lista por comprensión

```
[ x | x <- [ 2 .. 500 ], length [ y | y <- [ 2 .. x - 1 ], x `mod` y == 0 ] == 0 ]
```

   al aplicar a cada elemento de la lista 10 veces la función $h(x) = (f \circ g)$ donde

$$\begin{cases} f(x) = succ(x) = sucesor\ de\ x \\ g(x) = 4x \end{cases}$$

   ¿Cuál es el resultado de sumar todos los elementos de la lista retornada por *mapNTimes*?