Facultad de Ciencias - UNAM
Estructuras Discretas 2023-2
Ejercicios de repaso

Laura Friedberg Gojman
Saúl Adrián Rojas Reyes
José Manuel Madrigal Ramírez

17/Marzo/2023
Fecha de entrega: 26/Marzo/2023 a las 23:59 hrs

Introducción

Con los siguientes ejercicios vamos a repasar algunos conceptos vistos en prácticas anteriores. Puedes hacer estos ejercicios compartiendo ideas con otros compañeros y discutiendo las respuestas a las que cada uno haya llegado.

Escribe tus respuestas en el archivo `Ejercicios1.hs` que subí junto a este PDF, allí encontrarás las firmas de las funciones y una cláusula con un mensaje de error que deberás sustituir por tu código.

Fechas

Vamos a programar un conjunto de funciones que nos permitan verificar fechas válidas tomando en consideración años bisiestos.

1. Comencemos definiendo una función que nos indica si un número es divisible entre otro. *Pista: Investiga las funciones* `mod` *y* `rem` *¿Cuál de ellas nos puede servir?*

```
*Ejercicios2>divisible 9 3
True
*Ejercicios2>divisible 9 4
False
```

2. Considerando que un año bisiesto es aquel que:
   a) es divisible por 4 (por ejemplo 2008)
   b) excepto si es divisible por 100, entonces **no** es un año bisiesto
   c) aunque si es divisible por 400, entonces **si** es un año bisiesto

Definamos una función que nos regrese un Booleano e indique si un entero dado representa un año bisiesto o no:

```
*Ejercicios2>bisiesto 2008
True
*Ejercicios2>bisiesto 1900
False
*Ejercicios2>bisiesto 2000
True
*Ejercicios2>bisiesto 2007
False
```

3. Completa el tipo de dato `Mes` y hazlo pertenecer a la clase de tipo `Eq` y `Show`. *Pista: recuerda el uso de* `deriving`.

4. Define la función que nos regresa la cantidad de días que tuvo un mes dependiendo del año.

```
*Ejercicios2>diasDelMes Febrero 2008
29
*Ejercicios2>duasDelMes Febrero 2007
28
```

5. En el documento podemos encontrar el tipo de dato `Fecha`. Conviértelo en instancia de `Show` usando `deriving` y después define una función que indique si una fecha es válida:

```
*Ejercicios2>fechaValida (F 29 Febrero 2008)
True
*Ejercicios2>fechaValida (F 29 Febrero 2008)
False
*Ejercicios2>fechaValida (F 0 Marzo 2008)
False
```

Tarjetas de crédito

¿Alguna vez te preguntaste como hacían los sitios web para validar tu tarjeta de crédito cuando comprás en línea? No buscan en una base de datos gigantesca de números, tampoco usan magia. En realidad, la mayoría de los proveedores de tarjetas de crédito usan una fórmula de suma de control para distinguir números válidos de números aleatorios (o de errores de tipeo).

En esta sección, vamos a implementar el algoritmo de validación para tarjetas de crédito que sigue los siguientes pasos:

- Duplicar el valor de uno de cada dos dígitos, empezando desde la derecha. Es decir, el último dígito no cambia; el penúltimo es duplicado; el anterior no cambia; y sigue así. Por ejemplo, `[1,3,8,6]` se vuelve `[2,3,16,6]`.
- Sumar los dígitos de los valores duplicados y los dígitos no duplicados del número original. Por ejemplo, `[2,3,16,6]` se vuelve `2+3+1+6+6 = 18`.

- Calcular el residuo cuando esa suma es dividida por 10. En el ejemplo anterior, el residuo sería 8.
- Si el resultado es 0 entonces el número es válido.

1. Primero debemos descomponer un número en su último dígito y el resto del número. Dividiremos esta tarea en dos partes. En primer lugar definiremos una función que regrese el último dígito de un número dado:

```
*Ejercicios2>ultimoDigito 123
3
*Ejercicios2>ultimoDigito 0
0
```

2. Ahora definamos la función que regresa el número sin su último dígito:

```
*Ejercicios2>sinUltimo 123
12
*Ejercicios2>sinUltimo 5
0
```

3. Para la descomposición de un número en sus dígitos definiremos una función que dado un número, nos devuelva la lista de sus dígitos:

```
*Ejercicios2>aDigitos 1234
[1,2,3,4]
*Ejercicios2>aDigitos 0
[]
*Ejercicios2>aDigitos -23
[]
```

4. Hecho lo anterior, duplicaremos uno de cada dos definiendo la función `dupicaCadaDos`:

```
*Ejercicios2>duplicaCadaDos [8, 7, 6, 5]
[16, 7, 12, 5]
*Ejercicios2>duplicaCadaDos [1, 2, 3]
[1, 4, 3]
```

5. La salida de la función anterior es una mezcla de números de uno y dos dígitos. Vamos a definir una función que calcule la suma de todos los dígitos, en el siguiente ejemplo, observa que los números de dos cifras se *separan*, teniendo la suma `1 + 6 + 7 + 1 + 2 + 5 = 22`:

```
*Ejercicios2>sumaDigitos [16, 7, 12, 5]
22
```

6. Finalmente define la función que utilice todas las anteriores para indicar si un entero dado es un número de tarjeta válido:

```
*Ejercicios2>valida 4012888888881881
True
*Ejercicios2>valida 4012888888881882
False
```

Entrega

La entrega es **por parejas** y consistirá en un archivo comprimido que contenga:

- El archivo `Ejercicios1.hs` con las respuestas a los ejercicios.
- Un archivo ReadMe.txt que incluya sus nombres y algún comentario o idea sobre la práctica. Si trabajaste con un compañero para desarrollar las ideas incluye un párrafo describiendo la dinámica de su colaboración. Opiniones y sugerencias también son bien recibidos.

Sube el archivo comprimido a la plataforma Google Classroom incluyendo en un comentario a la persona con quien trabajaron y utilizando el siguiente formato para el nombre del archivo:

Ejercicios1_ Apellido1-Nombre1-Apellido2-Nombre2.zip

Cualquier duda que tengas no dudes en enviarme un correo. 😄