Introducción a Haskell (1)

Ejercicio 01: Realice las siguientes operaciones utilizando el menor número de paréntesis:

$$2 \cdot 3 + 5$$

$$2 + 2 \cdot 3 + 1$$

$$3^4 + 5 \cdot 2^5 + 1$$

Ejercicio 02: Cree una función `mult3 x` que devuelva True si la entrada es un múltiplo de 3 y en caso contrario False.

Ejercicio 03: Cree una función `mult5 x` que devuelva `True` si la entrada es un múltiplo de 5 y en caso contrario False.

Ejercicio 04: Cree una función `mult35 x` que devuelva `True` si la entrada es un múltiplo de 3 y 5 y en caso contrario False.

Ejercicio 05: Escriba un programa que devuelva `True` si la entrada es menor que -1 o (mayor que 1 Y un múltiplo de 2), y en caso contrario False.

Ejercicio 06: Cree una función que tome un tipo `Integer` y lo devuelva dividido por 2:

```
div2d :: Integer -> Double
```

Ejercicio 07: Cree una función que reciba un ángulo `a`y devuelva una tupla que contenga el seno de la mitad de ese ángulo usando la identidad:

$$sin(x/2) = \pm\sqrt{\frac{1-cos(x)}{2}}$$

Ejercicio 08: Cree una lista de años bisiestos desde el año 1 hasta el año actual.
Ejercicio 09: Encuentre los primeros 10 años bisiestos.
Ejercicio 09: Encuentre los últimos 10 años bisiestos (pista: use la función `length`para determinar la longitud de la lista).
Ejercicio 10: Cree una tupla donde el primer elemento tenga la mitad de los años bisiestos y el segundo elemento la otra mitad.
Ejercicio 11: Cree un concatenador de cadenas que concatene dos cadenas separadas por espacios.
Ejercicio 12: Dada la cadena “0123456789”, cree una lista con los dígitos en el formato `Integer`.

Introducción a Haskell (2)

Ejercicio 01: Cree una función `esTriangulo` que determine si tres lados x, y, z pueden formar un triángulo.
Ejercicio 02: Cree una función `tipoTriangulo` que determine el tipo de triángulo formado por los tres lados x, y, z.
Ejercicio 03: Implemente una función que realice una multiplicación etíope entre dos números.
Ejercicio 04: Cree una función que determine si un número es primo.
Ejercicio 05: Crea una función que calcule la suma de los dígitos de un número.
Ejercicio 06: Escribe una función que calcule la persistencia aditiva de un número.
Ejercicio 07: Haz una función que calcule el coeficiente binomial de (m, n).
Ejercicio 08: Crea una función que calcule el elemento (i, j) del triángulo de Pascal.

Introducción a Haskell (3)

Ejercicio 01: Cree una función `divisivel20 x` que devuelva verdadero si x es divisible por todos los números del 1 al 20.

Ejercicio 02: Cree una función `projectEuler5` que devuelva el primer número natural que regresa `True` a la función del ejercicio anterior. Piense en cómo reducir el costo computacional.

Ejercicio 03: Cree la lista de números de Fibonacci usando una función generadora.

Ejercicio 04: Utilizando la lista anterior, calcule la suma de los números pares de Fibonacci de los valores que no superen los 4.000.000. (Proyecto Euler 2)

Ejercicio 05: Cree una función para calcular el producto escalar entre dos vectores.

Ejercicio 06: Crea la función `collatz x` que devuelve $x \ / \ 2$, si x es par y $\left(\ 3 \ x + 1 \ \right)$ si es impar.

Ejercicio 07: Implementar una función `collatzLen x` que devuelva el tamaño de la lista formada por la aplicación repetida de `collatz` sobre el valor `x` hasta llegar al número 1.

Ejercicio 08: Encuentre el número `x` entre 1 y 1,000,000 que tiene la secuencia de Collatz más larga. (Proyecto Euler 14)