

## **Practico N° 2 – Introducción a Haskell**

Previo a la realización de este práctico es necesario instalar y leer el material sobre Haskell.

1. Escriba en la línea de comandos, de a una en una las siguientes expresiones en Hugs, y cada vez pulse la tecla ENTER.
  - a)  $20 * 33$
  - b)  $2^{18}$
  - c)  $24 * (67 - 111)$
  - d)  $2^{(6+3)}$
  - e)  $73 / 5$
  - f) `div 73 5`
  - g) `mod 73 5`
  - h) `mod (-20) 3`
  - i) `div (-20) 3`
  - j) Deduzca el comportamiento de las funciones `div` y `mod`, predefinidas en Haskell
  - k) Pruebe sus propios ejemplos
2. Escriba y ejecute las siguientes expresiones en Hugs.
  - a) `max 7 9`
  - b) `min 2 (-8)`
3. En Hugs no se pueden realizar definiciones en línea de comandos, para ello se deberá utilizar un archivo.hs. Pruebe ejecutar la siguiente línea en Hugs:  
`y = x+1`  
Cree un archivo en la carpeta Mis Documentos de su disco y denomínelo primero.hs  
Copie las siguientes líneas en su archivo y guarde.  
`y = x+1`  
`x = 2*3`  
Ejecutar en Hugs `File/open/` y luego busque `primero.hs` y selecciónelo.  
Escriba `x` en la línea de comandos y dé enter, lo mismo con `y`.  
Idem con  
`y = x^2`  
`x = (2*3)+(4-5)`
4. Calcular en Haskell:
  - a) `(8>2) || (4>1)`
  - b) `(8>2) && (4>1)`
  - c) `(8>2) && (4<1)`
  - d) Escribir y probar en Hugs sus propias expresiones utilizando los operadores lógicos introducidos.
5. Escriba y evalúe las siguientes expresiones en Hugs.
  - a) `7<0`
  - b) `8>2`
  - c) `div 8 4 == 2`
  - d) `div 4 3 == 0`
  - e) `mod 8 4 == 0`
  - f) `not (mod 8 4 == 7)`
  - g) `4 /= 4`
  - h) `div 8 9 /= 20`
  - i) `mod 50 10 <= 0`
  - j) `div 50 10 >= 6`
  - k) Agregue sus propias expresiones booleanas en Hugs y evalúelas

6. Realizar los siguientes cálculos en Hugs:
  - a) Calcular  $2^2$
  - b) Calcular  $2^4$
  - c) Calcular  $2^{32}$
  - d) Calcular  $2^{64}$
  - e) Calcular  $2^{200}$
  - f) Calcular  $2^4 :: \text{Int}$
  - g) Calcular  $2^{32} :: \text{Int}$
  - h) Calcular  $2^{32} :: \text{Integer}$
  - i) Calcular  $2^{31} :: \text{Int}$
  - j) Calcular  $2^{31} - 1 :: \text{Int}$
  - k) Calcular  $\text{minBound} :: \text{Int}$  y  $\text{maxBound} :: \text{Int}$  y comparar con los resultados de 9 y 10
  - l) Estudiar los resultados obtenidos y analizar posibles explicaciones
  
7. Verificar en Haskell:
  - a) `:type 'a'`
  - b) `:type "abcdf"`
  - c) `:type True`
  - d) `:type 4<5`
  - e) `:type "a"`
  - f) `length "perro"`
  - g) Deducir el tipo de `length`
  - h) Verificar `:type length`
  - i) Escribir y probar en Hugs sus propias expresiones
  
8. Probar en Haskell:
  - a) `(4, 'a', "gato")`
  - b) `:type (4, 'a', "gato")`
  - c) `:type ('a', "zapato")`
  - d) `:type ("haskell", "matematica")`
  - e) `:type (True, 5)`
  - f) `:type ('?', " pepito ")`
  - g) `:type ("cualquiera", (3.14, False))`
  
9. Probar en hugs:
  - a) `fst ("jota", (1, 'b'))`
  - b) `snd ("jota", (1, 'b'))`
  - c) Indicar el tipo de `fst` y `snd` y verificarlo en Haskell.
  
10. Abrir en el bloc de notas el archivo primero.hs  
Copiar al archivo primero.hs las siguientes definiciones de funciones:  
`f :: Int -> Int`  
`f n = 2^n`  
`g :: Int -> Integer`  
`g n = 2^n`  
Estudiar su significado.
  - a) Calcular `f 2`
  - b) Calcular `g 4`
  - c) Calcular `f 32`
  - d) Calcular `f 64`
  - e) Calcular `f 200`
  - f) Calcular `g 200`
  - g) Calcular `g 35`
  - h) Otras aplicaciones de `f` o `g` que deseen
  - i) Explicar los resultados.