Facultad de Ciencias Exactas, Ingeniería y Agrimensura
Escuela de Ciencias Exactas y Naturales
Departamento de Ciencia de la Computación
Estructuras de Datos y Algoritmos II

Práctica 2

Representación ASCII de los símbolos de la práctica

λ	$\leftrightarrow$	`\`	$\equiv$	$\leftrightarrow$	`==`	$\leqslant$	$\leftrightarrow$	`<=`
$\geqslant$	$\leftrightarrow$	`>=`	$\wedge$	$\leftrightarrow$	`&&`	$\circ$	$\leftrightarrow$	`.`
$+\!\!+$	$\leftrightarrow$	`++`	$\rightarrow$	$\leftrightarrow$	`->`	$\uparrow$	$\leftrightarrow$	`^`

1. Dar el tipo completo de las siguientes funciones:

a) *test* donde $test\ f\ x = f\ x \equiv x + 2$

b) *esMenor* donde $esMenor\ y\ z = y < z$

c) *eq* donde $eq\ a\ b = a \equiv b$

d) *showVal* donde $showVal\ x =$ `"Valor:"` $+\!\!+\ show\ x$

2. Dar el tipo de las siguientes operaciones y explicar su propósito:

a) $(+5)$

b) $(0<)$

c) (`'a'`:)

d) $(+\!\!+$`"\n"`$)$

e) *filter* $(\equiv 7)$

f) *map* $(+\!\!+[1])$

3. Dar al menos dos ejemplos de funciones que tengan el tipo indicado en cada caso:

a) $(Int \to Int) \to Int$

b) $Int \to (Int \to Int)$

c) $(Int \to Int) \to (Int \to Int)$

d) $Int \to Bool$

e) $Bool \to (Bool \to Bool)$

f) $(Int, Char) \to Bool$

g) $(Int, Int) \to Int$

h) $Int \to (Int, Int)$

i) $a \to Bool$

j) $a \to a$

4. Indicar si cada una de las siguientes expresiones está o no bien formada. En caso de que lo esté determinar el valor que denota, en caso contrario especificar si el error es sintáctico o de tipos:

a) **if** *true* **then** *false* **else** *true* **where** $false = True; true = False$

b) **if if then then else else**

c) $False \equiv (5 \geqslant 4)$

d) $1 < 2 < 3$

e) $1 +$ **if** (`'a'` < `'z'`) **then** -1 **else** 0

f) **if** $fst\ p$ **then** $fst\ p$ **else** $snd\ p$ **where** $p = (True, 2)$

g) **if** $fst\ p$ **then** $fst\ p$ **else** $snd\ p$ **where** $p = (True, False)$

5. Reescribir cada una de las siguientes definiciones sin usar **let**, **where** o **if**:

a) $f\ x = \textbf{let}\ (y, z) = (x, x)\ \textbf{in}\ y$

b) $greater\ (x, y) = \textbf{if}\ x > y\ \textbf{then}\ True\ \textbf{else}\ False$

c) $f\ (x, y) = \textbf{let}\ z = x + y\ \textbf{in}\ g\ (z, y)\ \textbf{where}\ g\ (a, b) = a - b$

6. Pasar de notación Haskell a notación de funciones anónimas (llamada notación lambda),

a) *smallest*, definida por

$$\begin{aligned} smallest\ (x, y, z) \mid\ & x \leqslant y \wedge x \leqslant z = x \\ \mid\ & y \leqslant x \wedge y \leqslant z = y \\ \mid\ & z \leqslant x \wedge z \leqslant y = z \end{aligned}$$

b) $second\ x = \lambda x \to x$

c) *andThen*, definida por

$$\begin{aligned} andThen\ True\ y &= y \\ andThen\ False\ y &= False \end{aligned}$$

d) $twice\ f\ x = f\ (f\ x)$

e) $flip\ f\ x\ y = f\ y\ x$

f) $inc = (+1)$

7. Pasar de notación lambda a notación Haskell,

a) $iff = \lambda x \to \lambda y \to \textbf{if}\ x\ \textbf{then}\ not\ y\ \textbf{else}\ y$

b) $alpha = \lambda x \to x$

8. Suponiendo que f y g tienen los siguientes tipos

$$\begin{aligned} f &:: c \to d \\ g &:: a \to b \to c \end{aligned}$$

y sea h definida como

$$h\ x\ y = f\ (g\ x\ y)$$

Determinar el tipo de h e indicar cuáles de las siguientes definiciones de h son equivalentes a la dada:

$$\begin{aligned} h &= f \circ g \\ h\ x &= f \circ (g\ x) \\ h\ x\ y &= (f \circ g)\ x\ y \end{aligned}$$

Dar el tipo de la función $(\circ)$.

9. La función *zip3* zipea 3 listas. Dar una definición recursiva de la función y otra definición con el mismo tipo que utilice la función *zip*. ¿Qué ventajas y desventajas tiene cada definición?

10. Indicar bajo qué suposiciones tienen sentido las siguientes ecuaciones. Para aquellas que tengan sentido, indicar si son verdaderas y en caso de no serlo modificar su lado derecho para que resulten verdaderas:

a) $[[\,]] +\!\!+ xs = xs$

b) $[[\,]] +\!\!+ xs = [xs]$

c) $[[\,]] +\!\!+ xs = [\,] : xs$

d) $[[\,]] +\!\!+ xs = [[\,], xs]$

e) $[[\,]] +\!\!+ [xs] = [[\,], xs]$

f) $[[\,]] +\!\!+ [xs] = [xs]$

g) $[\,] +\!\!+ xs = [\,] : xs$

h) $[\,] +\!\!+ xs = xs$

i) $[xs] +\!\!+ [\,] = [xs]$

j) $[xs] +\!\!+ [xs] = [xs, xs]$

11. Inferir, de ser posible, los tipos de las siguientes funciones:
(puede suponer que $sqrt :: Float \to Float$)

a) $modulus = sqrt \circ sum \circ map\ (\uparrow 2)$

b) $vmod\ [\,] \quad = [\,]$
$vmod\ (v : vs) = modulus\ v : vmod\ vs$

12. Dado el siguiente tipo para representar números binarios:

type $NumBin = [Bool]$

donde el valor *False* representa el número 0 y *True* el 1. Definir las siguientes operaciones tomando como convención una representación *Little-Endian* (i.e. el primer elemento de las lista de dígitos es el dígito menos significativo del número representado).

a) suma binaria

b) producto binario

c) cociente y resto de la división por dos

13. Definir las siguientes funciones usando listas por comprensión:

a) *divisors*, que dado un entero positivo x devuelve la lista de los divisores de x (y la lista vacía si el entero no es positivo).

b) *matches*, que dados un entero x y una lista de enteros descarta de la lista los elementos distintos a x.

c) *cuadruplas*, que dado un natural n, devuelve las cuadruplas (a, b, c, d) con $0 < a, b, c, d, \leqslant n$ que cumplen $a \uparrow 2 + b \uparrow 2 = c \uparrow 2 + d \uparrow 2$.

d) *unique*, que dada una lista xs de enteros, devuelve la lista con los elementos no repetidos de xs.
Por ejemplo, $unique\ [1, 4, 2, 1, 3] = [1, 4, 2, 3]$.

14. El producto escalar de dos listas de enteros de igual longitud es la suma de los productos de los elementos sucesivos (misma posición) de ambas listas. Usando listas por comprensión defina una función *scalarproduct* que devuelva el producto escalar de dos listas.

Sugerencia: Usar las funciones *zip* y *sum*.