

**Matemática I – Segunda Prueba Parcial – 29 de octubre de 2010**  
**SOLUCIÓN**

**Comentarios:**

1. En todos los ejercicios no podrán utilizarse funciones auxiliares, salvo que se indique explícitamente lo contrario.
2. Las funciones predefinidas de Haskell podrán usarse solamente si se indica explícitamente.

**Ejercicio 1 (1,5 puntos)**

Sean  $f: A \rightarrow B$ ,  $g: A \rightarrow (A \rightarrow B) \rightarrow B$ ,  $h: (A \rightarrow A) \rightarrow A \rightarrow A$ ,  $j: A \rightarrow A$ ,  $p: (A \rightarrow B) \rightarrow B$ ,  $x: A$ ,  $y: B$   
Indicar si las siguientes expresiones tienen tipo, en caso afirmativo indicarlo, justificando en todo caso la respuesta.

- a)  $g\ x$       b)  $g\ x\ x$       c)  $g\ x\ (f\ x)$       d)  $g\ x\ f$       e)  $p\ f$       f)  $p\ (g\ x)$

**Solución:**

- a) La función  $g$  recibe un parámetro de tipo  $A$  y una función de tipo  $(A \rightarrow B)$  en ese orden, por lo cual como  $x$  es de tipo  $A$ ,  $g\ x$  es de tipo  $(A \rightarrow B) \rightarrow B$ .
- b) La aplicación de funciones, por el contrario del implica, asocia hacia la izquierda, por lo tanto las siguientes expresiones  $(g\ x\ x)$  y  $((g\ x)\ x)$  son equivalentes. Deducimos que si  $(g\ x): (A \rightarrow B) \rightarrow B$ , entonces la expresión  $((g\ x)\ x)$  no tiene tipo (es incorrecta).
- c) La función  $f$  es de tipo  $(A \rightarrow B)$ , por lo cual  $(f\ x): B$  y volviendo al caso anteriormente demostrado,  $g\ x\ (f\ x)$  es incorrecta (no tiene tipo) pues  $(g\ x)$  espera un parámetro de tipo  $(A \rightarrow B)$  y está recibiendo uno de tipo  $B$ .
- d) La función  $g$  recibe un dato de tipo  $A$  (en este ejercicio es  $x$ ) y otro de tipo  $(A \rightarrow B)$  en este ejercicio es la función  $f$ , por lo cual  $(g\ x\ f)$  es de tipo  $B$ .
- e) La función  $p$  espera un dato de tipo  $(A \rightarrow B)$  y justamente  $f$  tiene este tipo, por lo cual  $(p\ f)$  es de tipo  $B$ .
- f) El parámetro  $(g\ x)$  es de tipo  $(A \rightarrow B) \rightarrow B$  y  $p$  recibe un dato  $A \rightarrow B$  por lo cual  $(p\ (g\ x))$  no tiene tipo.

**Ejercicio 2 (2,5 puntos)**

- a) Definir en Haskell una función recursiva **sumcuad** que recibe un entero  $x \geq 1$  y devuelve la suma de los cuadrados de los enteros desde 1 hasta  $x$ . Indicar su tipo.  
Ejemplo: `sumcuad 5 = 55` (observar que  $1+4+9+16+25 = 55$ ).
- b) Dar la secuencia de cómputo para `(sumcuad 4)`.

**Solución:**

- a) Definición de la función **sumcuad** cuyo tipo es  $(Integer \rightarrow Integer)$

```
sumcuad :: Integer -> Integer
sumcuad 0 = 0
sumcuad x = x^2 + sumcuad (x-1)
```

- b)  $\text{sumcuad } 4 = 4^2 + \text{sumcuad } 3 = 4^2 + (3^2 + \text{sumcuad } 2) =$   
 $= 4^2 + (3^2 + (2^2 + \text{sumcuad } 1)) = 4^2 + (3^2 + (2^2 + (1^2 + \text{sumcuad } 0))) =$   
 $= 4^2 + (3^2 + (2^2 + (1^2 + 0))) = 4^2 + (3^2 + (2^2 + 1)) =$   
 $= 4^2 + (3^2 + 5) = 4^2 + 14 = 16 + 14 = 30$

### Ejercicio 3 (2,5 puntos)

- a) Definir por casos la función **sumaplus** que recibe dos naturales (N) y devuelve el número natural siguiente a la suma de ambos (Indicar su tipo).  
Ejemplo: `sumaplus (S(S Z)) (S Z) = S(S(S(S Z)))`
- b) Dar la secuencia de cómputo para `(sumaplus (S(S Z)) (S(S Z)))`.

### Solución:

- a) Definición de la función **sumaplus** cuyo tipo es  $(N \rightarrow N \rightarrow N)$

```
sumaplus :: N → N → N
sumaplus Z Z = (S Z)
sumaplus Z (S m) = (S (S m))
sumaplus (S n) Z = (S (S n))
sumaplus (S n) (S m) = S (sumaplus n (S m))
```

Ejemplo de aplicación:

```
Parcial2> sumaplus 4 2
S (S (S (S (S (S (S Z))))))
Parcial2> toInteger (S (S (S (S (S (S (S Z))))))
7
Parcial2> sumaplus 2 5
S (S (S (S (S (S (S Z))))))
Parcial2> toInteger (S (S (S (S (S (S (S Z))))))
8
Parcial2> sumaplus 1 1
S (S (S Z))
Parcial2> sumaplus 5 5
S (S (S (S (S (S (S (S (S (S (S Z))))))))))
Parcial2> toInteger (S (S (S (S (S (S (S (S (S (S (S Z))))))))))
11
```

- b) `sumaplus (S (S Z)) (S (S Z)) = S (sumaplus (S Z) (S (S Z))) =`  
`= S (S (sumaplus Z (S (S Z)))) = S (S (S (S (S Z))))`

### Ejercicio 4 (2,5 puntos)

Aquí pueden utilizarse **funciones auxiliares** y **funciones predefinidas** de Haskell !!!

- a) Definir una función **divlista** cuyo tipo sea el siguiente:  $Integer \rightarrow [a] \rightarrow ([a], [a])$ , que dados un entero positivo y una lista, divida la lista en el elemento enésimo.  
Ejemplo: `divlista 3 [-4,5,8,-3,7,0,1] = ([-4,5,8], [-3,7,0,1])`
- b) Dar la secuencia de cómputo para `(divlista 2 [-5,0,1,3,7])`

### Solución:

- a) Definición de función **divlista** cuyo tipo es  $Integer \rightarrow [a] \rightarrow ([a], [a])$ .

```
largo :: [a] → Integer
largo [] = 0
largo (x:xs) = 1 + largo(xs)
```

```
divlista :: Integer → [a] → ([a], [a])
divlista _ [] = ([],[a])
divlista 0 (x:p) = ([],[x:p])
divlista 1 (x:p) = ([x],p)
divlista n (x:p) = if (largo (x:p) > n) then
    (x:(fst (divlista (n-1) p)),snd (divlista (n-1) p))
    else ((x:p),[])
```

$$\begin{aligned} \text{b) } \text{divlista } 2 \ [-5,0,1,3,7] &= (-5:(\text{fst} (\text{divlista } 1 \ [0,1,3,7])), \text{snd} (\text{divlista } 1 \ [0,1,3,7])) = \\ &= (-5:(\text{fst} ([0],[1,3,7])), \text{snd} ([0], [1,3,7])) = \\ &= (-5:([0]), [1,3,7]) = ([-5,0], [1,3,7]) = \end{aligned}$$

### Ejercicio 5 (3 puntos)

a) En Haskell definimos una función **matuno** de la siguiente forma, :

```
matuno :: [Integer]->[Integer]->[Integer]
matuno [] [] = []
matuno (a:x) [] = (a:x)
matuno [] (b:y) = (b:y)
matuno (a:x) (b:y) | (a<=b) = a:(matuno x (b:y))
                   | otherwise = b:(matuno (a:x) y)
```

¿Cuál es el comportamiento dicha función?

- b) ¿Cuál es el resultado si en la línea de comandos digitamos `matuno [-7,-3,2] [-10,1,8]`? Detalle la secuencia de cálculos para esta entrada.
- c) Probar por inducción que:  $\forall l, m \in \text{List } (A)$ , se cumple:  
`largo(matuno l m) = largo l + largo m`

### Solución:

a) La función **matuno** recibe dos listas p y q, de enteros y las asume ordenadas en forma creciente, pues en otro caso, se debería comparar el primer elemento de p con todos los elementos de q, cosa que no hace, sino que solo lo compara con el primer elemento de q. Devuelve una lista de enteros también ordenada en forma creciente cuyo resultado es el de mezclar los elementos de las dos listas.

$$\begin{aligned} \text{b) } \text{matuno } [-7,-3,2] \ [-10,1,8] &= -10:(\text{matuno } [-7,-3,2] \ [1,8]) = \\ &= -10:(-7:(\text{matuno } [-3,2] \ [1,8])) = -10:(-7:(-3:(\text{matuno } [2] \ [1,8]))) = \\ &= -10:(-7:(-3:(1:(\text{matuno } [2] \ [8]))) = -10:(-7:(-3:(1:(2:(\text{matuno } [] \ [8]))) = \\ &= -10:(-7:(-3:(1:(2:([8]))) = -10:(-7:(-3:(1:([2,8]))) = \\ &= -10:(-7:(-3:([1,2,8]))) = -10:(-7:([-3,1,2,8])) = -10:([-7,-3,1,2,8]) = \\ &= [-10,-7,-3,1,2,8] \end{aligned}$$

c)  $(\forall l, q \in \text{List Integer}) (\text{largo}(\text{matuno } l \ q) = \text{largo } l + \text{largo } q)$

Paso base:  $\text{largo}(\text{matuno nil nil}) = (\text{largo nil}) = 0$   
 $(\text{largo nil})+(\text{largo nil}) = 0 + 0 = 0$

Paso inductivo:

Hipótesis inductiva:  $\text{largo} (\text{matuno } l \ q) = (\text{largo } l)+(\text{largo } q)$

Tesis inductiva:  $\text{largo} (\text{matuno } (\text{cons } a \ l) \ q) = (\text{largo } (\text{cons } a \ l))+(\text{largo } q)$

Prueba del paso inductivo: (supongamos  $a \leq b$ , donde b es el primer elemento de q)

$$\begin{aligned} \text{largo} (\text{matuno } (\text{cons } a \ l) \ q) &= \text{largo} (\text{cons } a \ (\text{matuno } l \ q)) = \\ &= 1 + \text{largo} (\text{matuno } l \ q) = 1 + (\text{largo } l)+(\text{largo } q) = \text{largo} (\text{cons } a \ \text{nil}) + (\text{largo } l)+(\text{largo } q) = \\ &= \text{largo} (\text{cons } a \ l) + \text{largo } q \end{aligned}$$

En caso que  $a > b$  se razona de forma análoga.