Programación declarativa **(30 de noviembre de 2010)**

Apellidos:

Nombre:

Grupo 2C

El algoritmo de la multiplicación rusa consiste en:

- Escribir los números (`A` y `B`) que se desea multiplicar en la parte superior de sendas columnas.
- Dividir `A` entre 2, sucesivamente, ignorando el resto, hasta llegar a la unidad. Escribir los resultados en la columna `A`.
- Multiplicar `B` por 2 tantas veces como veces se ha dividido `A` entre 2. Escribir los resultados sucesivos en la columna `B`.
- Sumar todos los números de la columna `B` que estén al lado de un número impar de la columna `A`. Éste es el resultado.

Ejemplos:

Para multiplicar 27 y 82:

A	B	
27	82	
13	164	
6	328	(no se suma)
3	656	
1	1312	

Resultado: $82 + 164 + 656 + 1312 = 2214$

Para multiplicar 5 y 5:

A	B	
5	5	
2	10	(no se suma)
1	20	

Resultado: $5 + 20 = 25$

Ejercicio 1 Define la función

```
multRusa :: Integral a => a -> a -> a
```

que implemente al algoritmo de la multiplicación rusa en Haskell.

Usaremos el siguiente tipo de dato para representar árboles binarios con elementos de tipo `a`.

```
data Arbol a = Hoja a | Nodo a (Arbol a) (Arbol a)
     deriving Show
```

Por ejemplo, los árboles binarios

vendrán representados por:

```
a1 ::  Arbol Int
a1 = Nodo 3 (Nodo 3 (Hoja 7) (Hoja 3)) (Hoja 0)
a2 ::  Arbol Char
a2 = Nodo 'c' (Hoja 'b') (Nodo 'a' (Hoja 'b') (Hoja 'b'))
```

Ejercicio 2 Define la funcion

```
internos ::  Arbol a -> [a]
```

tal que `(internos x)` devuelve una lista con el conjunto de elementos internos del árbol. Por ejemplo

```
internos a1 ==> [3,3]
internos a2 ==> "ca"
```

Ejercicio 3 Define el predicado

```
tieneRamaI ::  Eq a => Arbol a -> Bool
```

tal que `(tieneRamaI x)` se verifica si el árbol binario `x` contiene, al menos, una rama formada por elementos iguales. Por ejemplo:

```
tieneRamaI a1 ==> True
tieneRamaI a2 ==> False
```