Practica 4

Árboles

---

listaAArbol

Escribe la función **listaAArbol** que recibe una lista **xs** regresa un árbol binario que contenga todos los elementos de **xs**

```
listaAArbol [1, 2, 3, 4, 5]
AB 1 (AB 2 (H 4) (H 5)) (H 3)

listaAArbol [8, 5, 10, 4, 11, 6, 9]
AB 8 (AB 5 (AB 4 (H 6) (H 9)) (H 11)) (H 10)
```

minimo

Escribe la función **minimo** que recibe un árbol binario y obtiene el elemento minimo de este.

```
minimo (AB 8 (AB 5 (H 4) (H 6)) (AB 10 (H 9) (H 11)))
4

minimo (AB 1 (H (-1)) (AB 2 Vacio (AB 3 Vacio (H 4))))
-1
```

maximo

Escribe la función **maximo** que recibe un árbol binario y obtiene el elemento maximo de este.

```
maximo (AB 8 (AB 5 (H 4) (H 6)) (AB 10 (H 9) (H 11)))
11

maximo (AB 1 (H (-1)) (AB 2 Vacio (AB 3 Vacio (AB 4 Vacio (H 5)))))
5
```

inorder

Escribe la función **inorder** que toma un árbol y recorre sus elementos primero por su subárbol izquierdo, despues la raíz y al final su subárbol derecho, y devuelve una lista que contenga los elementos visitados en ese orden.

```
inorder (AB 8 (AB 5 (H 4) (H 6)) (AB 10 (H 9) (H 11)))
[4,5,6,8,9,10,11]

inorder  (AB 1 (H (-1)) (AB 2 Vacio (AB 3 Vacio (AB 4 Vacio (H 5)))))
```

```
[-1,1,2,3,4,5]
```

postorder

Escribe la función **postorder** que toma un árbol y recorre sus elementos primero por su subárbol izquierdo, al final su subárbol derecho y finalmente su raíz, y devuelve una lista que contenga los elementos visitados en ese orden.

```
postorder (AB 1 (H (-1)) (AB 2 Vacio (AB 3 Vacio (AB 4 Vacio (H 5)))))
[-1,5,4,3,2,1]

postorder (AB 8 (AB 5 (H 4) (H 6)) (AB 10 (H 9) (H 11)))
[4,6,5,9,11,10,8]
```

contarPor

Escribe la función **contarPor** que recibe una función que devuelve un booleano, un árbol y regresa el numero de elementos del árbol que al aplicarles la función esta devuelva True como resultado.

```
contarPor (> 3) (AB 6 (AB 3 (H 2) (H 4)) (AB 8 (H 7) (H 9)))
5

contarPor (\x -> (mod x 2) == 0) (AB 6 (AB 3 (H 2) (H 4)) (AB 8 (H 7) (H 9)))
4
```

listaPor

Escribe la función **contarPor** que recibe una función que devuelve un booleano, un árbol y regresa una lista que contenga los elementos del árbol que al aplicarles la función esta devuelva True como resultado.

```
listaPor (> 3) (AB 6 (AB 3 (H 2) (H 4)) (AB 8 (H 7) (H 9)))
[6,4,8,7,9]

listaPor (\x -> (mod x 2) == 0) (AB 6 (AB 3 (H 2) (H 4)) (AB 8 (H 7) (H 9)))
[6,2,4,8]
```

mapT

Escribe la función **mapT** que recibe una función, un árbol y regresa otro árbol cuyos elementos son el resultado de aplicar la función a cada uno de los elementos del árbol recibido como parametro.

```
mapT (+1) (AB 6 (AB 3 (H 2) (H 4)) (AB 9 (H 8) (H 10)))
AB 7 (AB 4 (H 3) (H 5)) (AB 10 (H 9) (H 11))

mapT show (AB 10 (AB 7 (H 6) (H 8)) (AB 13 (H 12) (H 14)))
AB "10" (AB "7" (H "6") (H "8")) (AB "13" (H "12") (H "14"))
```

filterT

Escribe la función **filterT** que recibe una función que devuelve un booleano, un árbol y regresa otro árbol cuyos elementos son aquellos del árbol recibido como parametro que cumplen que al aplicar la función sobre ellos esta devuelva True.

```
filterT (>9) (AB 10 (AB 7 (H 6) (H 8)) (AB 13 (H 12) (H 14)))
AB 10 Vacio (AB 13 (H 12) (H 14))

filterT (\x -> (mod x 2) == 0) (AB 6 (AB 3 (H 2) (H 4)) (AB 9 (H 8) (H 10)))
AB 6 (AB 4 (H 2) Vacio) (AB 10 (H 8) Vacio)
```

Para verificar sus funciones usen los archivos Practica4Test.hs. Una vez cargado dentro de ghci, deberan de ejecutar:

```
quickCheck <NombreDelTestAComprobar>
Por ejemplo:
Si quieren propar su función minimo escriben:
quickCheck prop_minimo
```