## G) Operaciones sobre árboles binarios

**Dar ecuaciones** que definan las siguientes operaciones sobre los tipos de datos Arbin Int o Arbin t según el caso:

### 1) nveces -- #

```
nveces :: (t, Arbin t) -> Int

nveces(w, Avacio) = 0
nveces(w, Crear(x, a, b))
 | w /= x                    = nveces(w, a) + nveces(w, b)
 | w == x                    = 1 + nveces(w, a) + nveces(w, b)
```

### 2) frontera -- #

```
frontera :: (Arbin t) -> [t]

frontera(Avacio) = []
frontera(Crear(x, a, b))
 | es_vacio(a) && es_vacio(b)    = x:[]
 | otherwise                     = frontera(a) ++ frontera(b)
```

### 3) ninternos -- #

```
ninternos :: (Arbin t) -> Int

ninternos(Avacio) = 0
ninternos(Crear(x, a, b))
 | es_vacio(a) && es_vacio(b)    = 0
 | otherwise                     = 1 + ninternos(a) + ninternos(b)
```

### 4) nhojas -- #

```
nhojas :: (Arbin t) -> Int

nhojas(Avacio) = 0
nhojas(Crear(x, a, b))
 | es_vacio(a) && es_vacio(b)    = 1
 | otherwise                     = nhojas(a) + nhojas(b)
```

## 5) nodos_nivel -- #

```
nodos_nivel :: (Int, Arbin t) -> Int

nodos_nivel(niv, Avacio)
 | niv ≤ 0                    = error "Nivel inadecuado"
 | otherwise                  = 0

nodos_nivel(niv, Crear(x, a, b))
 | niv ≤ 0                    = error "Nivel inadecuado"
 | niv == 1                   = 1
 | otherwise                  = nodos_nivel(niv – 1, a) +
                                    nodos_nivel(niv – 1, b)
```

## 6) listanivel -- #

```
listanivel :: (Int, Arbin t) -> [t]

listanivel(niv, Avacio)
 | niv ≤ 0                    = error "Nivel inadecuado"
 | otherwise                  = []

listanivel(niv, Crear(x, a, b))
 | niv ≤ 0                    = error "Nivel inadecuado"
 | niv == 1                   = x:[]
 | otherwise                  = listanivel(niv – 1, a) ++
                                    listanivel(niv – 1, b)
```

## 7) es_rama -- #

```
es_rama :: ([t], Arbin t) -> Bool

es_rama([], a)
 | es_vacio(a)                = True
 | otherwise                  = False

es_rama(w:s, a)
 | es_vacio(a)                = False
 | w /= raiz(a)               = False
 | otherwise                  = es_rama(s, izqdo(a)) || es_rama (s, drcho(a))
```

Siempre que hay dos parámetros de los nuevos tipos de datos (dos listas, dos árboles, una lista y un árbol, etc.) sólo hacemos la distinción de vacío (o vacía) y no vacío (o no vacía) para uno de ellos. Por eso en este caso hemos distinguido expresamente la lista vacía y la no vacía mientras que el árbol binario aparece como una variable (sin especificar expresamente si es vacío o no).

## 8) es_espejo -- #

```
es_espejo :: (Arbin t, Arbin t) -> Bool

es_espejo(Avacio, a)
 | es_vacio(a)              = True
 | otherwise                = False

es_espejo(Crear(x, b, c), a)
 | es_vacio(a)              = False
 | x /= raiz(a)             = False
 | otherwise                = es_espejo(b, drcho(a)) && es_espejo(c, izqdo(a))
```

Siempre que hay dos parámetros de los nuevos tipos de datos (dos listas, dos árboles, una lista y un árbol, etc.) sólo hacemos la distinción de vacío (o vacía) y no vacío (o no vacía) para uno de ellos. Por eso en este caso hemos distinguido expresamente los casos de árbol vacío y no vacío para el primer parámetro mientras que el segundo árbol binario aparece como una variable (sin especificar expresamente si es vacío o no).

## 9) es_prefijo -- #

```
es_prefijo :: (Arbin t, Arbin t) -> Bool

es_prefijo(Avacio, a) = True

es_prefijo(Crear(x, b, c), a)
 | es_vacio(a)          = False
 | x /= raiz(a)         = False
 | otherwise            = es_prefijo(b, izqdo(a)) && es_prefijo(c, drcho(a))
```

Siempre que hay dos parámetros de los nuevos tipos de datos (dos listas, dos árboles, una lista y un árbol, etc.) sólo hacemos la distinción de vacío (o vacía) y no vacío (o no vacía) para uno de ellos. Por eso en este caso hemos distinguido expresamente los casos de árbol vacío y no vacío para el primer parámetro mientras que el segundo árbol binario aparece como una variable (sin especificar expresamente si es vacío o no).

### 10) es_subarbol -- #

```
es_subarbol :: (Arbin t, Arbin t) -> Bool

es_subarbol(Avacio, a) = True

es_subarbol(Crear(x, b, c), a)
 | es_vacio(a)                    = False
 | es_prefijo(Crear(x, b, c), a)  = True
 | otherwise                      = es_subarbol(Crear(x, b, c), izqdo(a)) ||
                                    es_subarbol(Crear(x, b, c), drcho(a))
```

Siempre que hay dos parámetros de los nuevos tipos de datos (dos listas, dos árboles, una lista y un árbol, etc.) sólo hacemos la distinción de vacío (o vacía) y no vacío (o no vacía) para uno de ellos. Por eso en este caso hemos distinguido expresamente los casos de árbol vacío y no vacío para el primer parámetro mientras que el segundo árbol binario aparece como una variable (sin especificar expresamente si es vacío o no).

### 11) preorden -- #

```
preorden :: (Arbin t) -> [t]

preorden(Avacio) = []

preorden(Crear(x, a, b)) = (x:[]) ++ preorden(a) ++ preorden(b)
```

### 12) inorden -- #

```
inorden :: (Arbin t) -> [t]

inorden(Avacio) = []

inorden(Crear(x, a, b)) = inorden(a) ++ (x:[]) ++ inorden(b)
```

### 13) postorden -- #

```
postorden :: (Arbin t) -> [t]

postorden(Avacio) = []

postorden(Crear(x, a, b)) = postorden(a) ++ postorden(b) ++ (x:[])
```