VIDEO EXPLICATIVO VIDEO

INTRO Sabemos que con sólo el orden previo de un árbol no podemos reconstruir el mismo. Sin embargo, si agregamos información adicional, si lo podemos hacer. En este caso agregaremos para cada elemento del árbol un valor que indica el tipo de nodo, a saber

- `INT`: nodo INTERIOR
- `HOJ`: nodo que es HOJA
- `LAM`: nodo LAMBDA (no dereferenciable)

Entonces podemos serializar un árbol de `n` nodos como una lista de `2*n` valores, donde cada par de valores representa el tipo de nodo y la etiqueta del mismo. Es decir la lista contiene `[TIPO1,VALOR1,TIPO2,VALOR2,TIPO3,VALOR3...]`

- **NOTA1:** En el caso de los nodos Lambda por supuesto el valor en la lista e irrelevante. En los ejemplos se ha agregado 0.
- **NOTA2:** Para los tipos `INT`,`HOJ`,`LAM` se utilizan valores de tipo `const int` definidos al principio del `program.cpp`.

EJEMPLOS

```
L:[INT,0,HOJ,1,HOJ,2]               => B:(0 1 2)
L:[INT,0,HOJ,1,LAM,0]               => B:(0 1 .)
L:[INT,0,LAM,0,HOJ,1]               => B:(0 . 1)
L:[INT,0,HOJ,1,INT,2,LAM,20,HOJ,3] => B:(0 1 (2 . 3))
L:[INT,0,INT,7,INT,1,LAM,0,HOJ,8,LAM,0,HOJ,5]
                                    => B:(0 (7 (1 . 8) .) 5)
```

CONSIGNA Escribir una función

```
void list2btree(list<int> &L,btree<int> &B);
```

que, dado una lista `L` genera un árbol binario `B` de acuerdo a la convención definida arriba.

AYUDA

- Utilizar una estrategia recursiva, con una función auxiliar.
- La función va recorriendo el árbol en orden previo y extrayendo pares de valores de la lista (borrando) e insertando nodos en el árbol
- Para un nodo dado extrae dos valores (TIPO y VALOR) de la lista (borrando!).
- Si el tipo es `LAM` entonces no hay que hacer nada.
- Caso contrario inserta el valor en el nodo.
- Si el tipo es `HOJ` entonces ya está.
- Si es `INT` entonces llama recursivamente sobre el hijo derecho y el hijo izquierdo.

ZIP Enlace al zip