Video explicativo

CONSIGNA

Escribir una función `void abbpred(btree<int> &T,comp_t comp);` que, dado un árbol binario `T`, extrae todos sus elementos en orden previo y los vuelve a introducir de acuerdo a las reglas de un árbol binario de búsqueda (ABB), con respecto a la función de comparación (es un predicado binario) `comp`.

Ejemplos:

```
T:(1 3 2),   comp:"menor"    => T:(1 . (3 2 .))
T:(1 2 3),   comp:"menor"    => T:(1 . (2 . 3))
T:(2 1 3),   comp:"menor"    => T:(2 1 3)
T:(2 3 1),   comp:"mayor"    => T:(2 3 1)
T:(1 2 3),   comp:"mayor"    => T:(1 (2 3 .) .)
T:(2 1 3),   comp:"mayor"    => T:(2 3 1)
T:(1 -3 -2), comp:menor_abs  => T:(1 . (-3 -2 .))
T:(-1 2 -3), comp:menor_abs  => T:(-1 . (2 . -3))
```

Las funciones de comparación son por mayor y menor y por menor en valor absoluto. Pero por supuesto el código debería funcionar para cualquier función de comparación.

```
bool menor(int x,int y) { return x<y; }
bool mayor(int x,int y) { return x>y; }
bool menor_abs(int x,int y) { return abs(x)<abs(y); }
```

AYUDA

- Extraer todos los elementos del árbol **en orden previo** a una list `L`.
- Eliminar todos los elementos del árbol.
- Recorrer todos los elementos de la lista e ir insertándolos en el árbol usando las reglas de ABB, es decir, si el elemento es menor bajar por el hijo izquierdo, si no bajar por el derecho, hasta llegar a un `Lambda`, e insertar allí.
- Tener en cuenta que para comparar deben usar la función `comp`; no deben usar el operador `<`. Es decir cuando van bajando por un nodo del árbol `n` para comparar deben hacer `comp(x,*n)`, si da verdadero bajan por izquierda, caso contrario por derecha.

ENLACE PARA EL ZIP

ENLACE