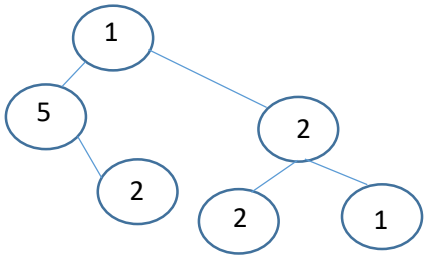
Árboles zurdos

Un árbol binario de enteros es *zurdo* cuando no es vacío, y la diferencia entre la suma de los valores en los nodos de su hijo izquierdo y la suma de los valores en los nodos de su hijo derecho es, al menos, la altura del árbol menos 1. Por ejemplo, el siguiente árbol es un árbol *zurdo*:



Efectivamente, (i) la suma de los valores en su hijo izquierdo es 7; (ii) la suma de los valores en su hijo derecho es 5; (iii) la altura del árbol es 3. Por tanto, 7-5 = 2 ≥ 3 -1 = 2.

En este control hay que diseñar e implementar un algoritmo que, dado un árbol binario de enteros, devuelva *true* si se trata de un árbol zurdo, y *false* en otro caso.

Trabajo a realizar

Se debe completar un programa que lee una serie de filas, cada una representando un árbol binario de enteros, e imprima por la salida `ZURDO` en caso de que el árbol sea zurdo, o `NO_ZURDO` en caso de que no lo sea. Para ello, se proporciona un archivo `main.cpp` en el que se implementa la lógica de entrada / salida necesaria. El código proporcionado no debe modificarse. Lo único que hay que hacer es implementar la función

```
bool es_zurdo(Arbin<int> a);
```

Dicha función debe devolver *true* si *a* es *zurdo*, y *false* en caso contrario. Aparte de esta función, podrán añadirse todas aquellas funciones auxiliares que se consideren necesarias.

Para probar el programa debe tenerse en cuenta que los árboles se codifican en la entrada de acuerdo con el siguiente criterio:

- El árbol vacío se representa como `#`
- Un árbol simple con raíz *A* se representa como [*A*]
- Un árbol compuesto con raíz A se representa como (τ_i A τ_d), donde τ_i es la representación del hijo izquierdo, y τ_d la del derecho.

De esta forma, el árbol mostrado anteriormente se codificará como:

```
((#5[2])1([2]2[1]))
```

Ejemplo de entrada / salida:

Entrada	Salida
`#`	`NO_ZURDO`
`[5]`	`ZURDO`
`((#5[2])1([2]2[1]))`	`ZURDO`
`(([2]1[3])5[17])`	`NO_ZURDO`