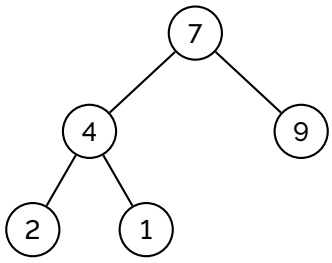
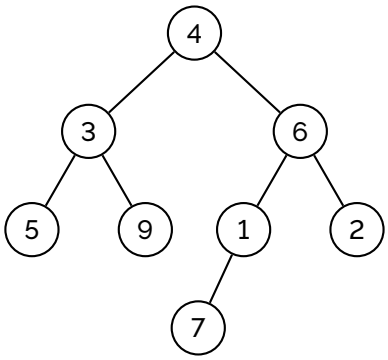
# Árboles estables en altura

Decimos que un árbol binario es *estable en altura* si el árbol nunca disminuye de altura cuando quitamos alguna de sus hojas. Por ejemplo:




El árbol de la izquierda es estable en altura, ya que si quitamos cualquiera de las hojas (2, 1 o 9), el árbol sigue teniendo altura 2. Por el contrario, el árbol de la derecha no lo es, ya que al quitar la hoja que contiene el 7, la altura del árbol disminuye.

Por convenio, suponemos que el árbol vacío es estable en altura.

Se pide:

1. Implementar una función `estable_altura` con la siguiente cabecera:

   ```
   bool estable_altura(const BinTree<T> &arbol);
   ```

   Esta función debe devolver `true` si el árbol pasado como parámetro es estable en altura, o `false` en caso contrario. Puedes crear las funciones auxiliares que sean necesarias.

2. Indica el coste de la función anterior, en función del número de nodos del árbol de entrada.

## Entrada

La entrada comienza con un número que indica el número de casos de prueba que vienen a continuación. Cada caso de prueba consiste en una línea con la descripción de un árbol binario mediante la notación vista en clase. El árbol vacío se representa mediante `.` y el árbol no vacío mediante `(iz x dr)`, siendo `x` la raíz, e `iz` y `dr` las representaciones de ambos hijos.

## Salida

Para cada caso de prueba debe imprimirse `SI` (sin tilde) si el árbol es estable en altura, o `NO` en caso contrario.

## Entrada de ejemplo

```
4
(((. 2 .) 4 (. 1 .)) 7 (. 9 .))
(((. 5 .) 3 (. 9 .)) 4 (((. 7 .) 1 .) 6 (. 2 .)))
((. 1 .) 5 .)
.
```

## Salida de ejemplo

```
SI
NO
NO
SI
```

## Autor

Manuel Montenegro