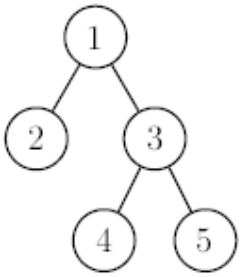
Hojas profundas

En este control hay que implementar una función que, dado un árbol binario de enteros y un número entero no negativo *k*, determine el número de hojas cuya profundidad es mayor que *k*. Por ejemplo, para el siguiente árbol



la función devolvería 3 si k=0 o k=1, devolvería 2 si k=2 y devolvería 0 si k=3.

Trabajo a realizar

Se debe completar un programa que lea una serie de filas, cada una representando un árbol binario de enteros y un número *k*, e imprima por la salida el número de hojas del árbol cuya profundidad es mayor que *k*. Se proporciona el archivo `main.cpp` en el que se implementa la lógica de entrada / salida necesaria. El código proporcionado no debe modificarse, salvo lo que corresponda a la implementación de la siguiente función

```
// Devuelve el número de hojas del árbol recibido cuya profundidad es mayor que un valor
// dado como parámetro.
// Parámetros:
//     a: árbol binario de enteros.
//     k: profundidad a superar para que una hoja sea contabilizada.
// Resultado:
//     N° de hojas de a con profundidad mayor que k.
unsigned int numero_hojas_mas_profundas_que(const Arbin<int>& a, unsigned int k);
```

así como la incorporación de 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:

```
([2]1([4]3[5]))
```

Ejemplo de entrada / salida:

Entrada	Salida
([2]1([4]3[5])) 0	3
([2]1([4]3[5])) 1	3
([2]1([4]3[5])) 2	2
([2]1([4]3[5])) 3	0
([2]1([4]3[5])) 4	0
([2]1([4]3[5])) 100	0