Auxiliar 3: Recursión

Todos los problemas deben ser resueltos en *Python*, utilizando estrictamente la Receta de Diseño entregada a lo largo del curso. Use nombres apropiados para funciones y variables, y testee cada vez que sea posible.

1. Pensando en recursión

En esta parte, usted no deberá programar (siéntase feliz o triste, dependiendo de sus preferencias)

- Imagine que tiene un árbol binario como el de la figura 1. Como puede notar, un árbol es una estructura que está compuesta por *nodos*, cada uno de los cuales tiene un valor, un hijo izquierdo y un hijo derecho. Se le llama *hojas* a los nodos cuyos hijos izquierdo y derecho son nulos (es decir, los nodos que no tienen hijos). Del mismo modo, se le llama *raíz* al nodo inicial (el de "más arriba")
- Piense en cómo se puede dibujar (o recorrer) este árbol.
- Vea la pizarra :)

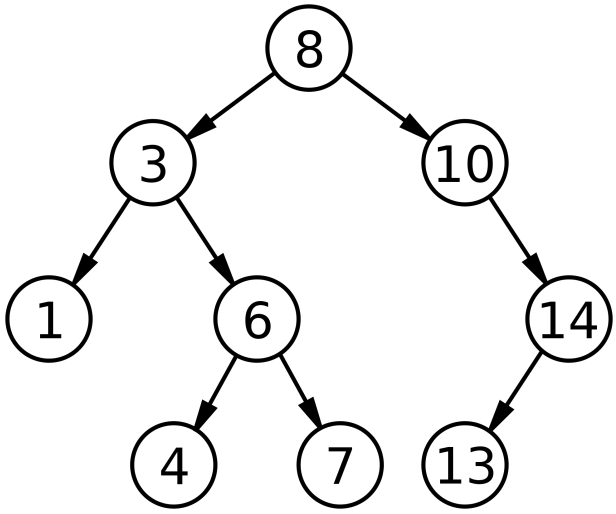

Figura 1: Árbol binario

2. Programando recursión

2.1. Pregunta 1

La multiplicación como la conocemos se puede pensar como una suma repetida varias veces. Implemente una función recursiva llamada `multiplicacion`, que reciba dos enteros y retorne la multiplicación de ambos.

Ejemplo: `multiplicacion(3,4)` entrega `12`
OJO: No puede usar los operadores *, /, %

2.2. Pregunta 2

Del mismo modo, la división entera también puede ser pensada recursivamente. Implemente la función `division`, que reciba dos enteros `a` y `b`, y retorne el valor de `a/b` (recuerde, división entera).

Ejemplos: `division(18,6)` entrega `3`; `division(11,4)` entrega `2`
OJO: Aquí tampoco puede usar los operadores *, /, %

2.3. Pregunta 3

Un palíndromo es una palabra que se lee igual de izquierda a derecha que de derecha a izquierda (como *oso, reconocer*, etc). Programe una función llamada `esPalindromo` que reciba un string y entregue True si la palabra es palíndromo, y False en caso contrario.

Ejemplo: `esPalindromo(‘‘reconocer’’)` entrega True