PARCIAL PRÁCTICO MÓDULO I - RECUPERATORIO

Implemente la clase **Parcial**, y el método:

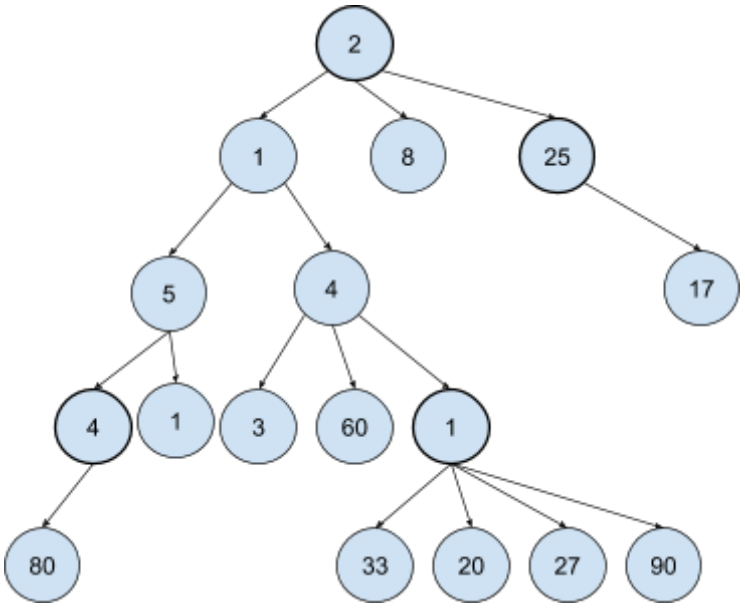
resolver(ArbolGeneral<Integer> arbol, int minimo): ListaGenerica<Integer>

que dado un árbol de números y un valor numérico, devuelve todos aquellos valores de nodos donde la suma de sus hijos supera a *mínimo*.

Tenga en cuenta las siguientes condiciones:

- La clase no cuenta con variables de instancia
- Debe realizar un recorrido **postorden**
- No puede recorrer 2 veces la estructura de aŕbol general
- La lista resultante no debe contener valores repetidos (no se puede generar una lista con todos los valores resultantes de procesar el árbol y después eliminar todos los duplicados)
- En el caso de las hojas, donde no tienen hijos, la suma da 0.

Ejemplo:



para el árbol dado, si *mínimo* es igual a 15, la lista resultante debería ser: [4, 1, 25, 2]. Notar que si bien el nodo con valor 4 cumple 2 veces, sólo aparece una vez en la lista resultante.

PARCIAL PRÁCTICO MÓDULO I - RECUPERATORIO

Implemente la clase **Parcial**, y el método:

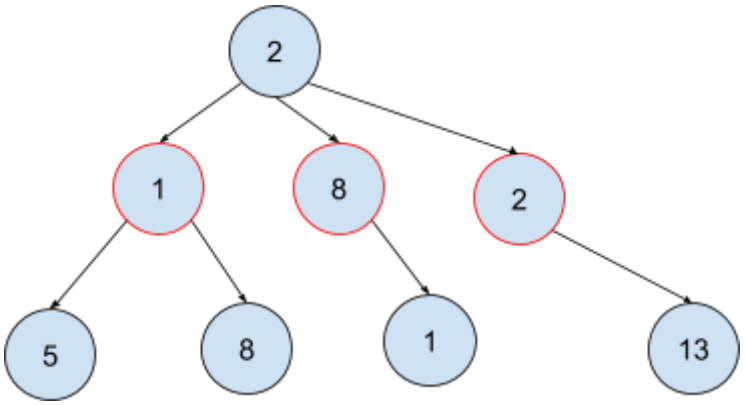
resolver(ArbolGeneral<Integer> arbol, int *nivelExcluido*): ?

que suma cada camino existente en el árbol, desde la raiz hasta una hoja, excluyendo de la suma los valores que se encuentren en el nivel *nivelExcluido* y devuelve cuantos resolvieron a un valor par o impar.

Tenga en cuenta las siguientes condiciones:

- La clase no cuenta con variables de instancia
- Debe realizar un recorrido **preorden**
- No puede recorrer 2 veces la estructura de arbol general
- La estructura devuelta debe respetar el paradigma POO
- La cantidad de niveles del árbol seguro es mayor que el valor *nivelExcluido*

Ejemplo:



para el árbol dado, si el *nivelExcluido es 1*, el valor resultante debería ser 3 caminos que suman valores impares y 1 camino que suma valores pares.

PARCIAL PRÁCTICO MÓDULO I - RECUPERATORIO

Implemente la clase **Parcial**, y el método:

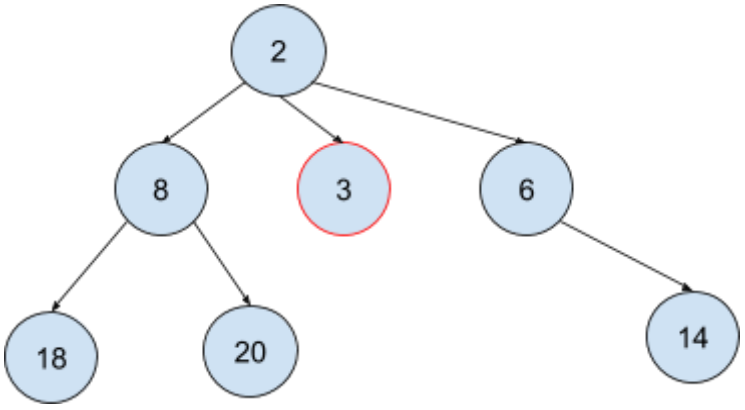
resolver(ArbolGeneral<Integer> arbol): ListaGenerica<Integer>

que dado un árbol general verifica si para cada nodo padre, cada uno de sus hijos es mayor o igual que el doble de su valor y a su vez, los va almacenando en una lista. En cuanto no se cumpla la condición, el algoritmo debe terminar.

Tenga en cuenta las siguientes condiciones:

- La clase no cuenta con variables de instancia
- Debe realizar un recorrido **preorden** hasta la condición de finalización (cuando no se cumple la condición)
- No puede recorrer 2 veces la estructura de árbol general

Ejemplo:



para el árbol dado, la lista resultante debe contener los valores: [2,8,18, 20]. El valor 3 no cumple el criterio, por tanto en ese momento el algoritmo termina y no se sigue procesando el árbol.