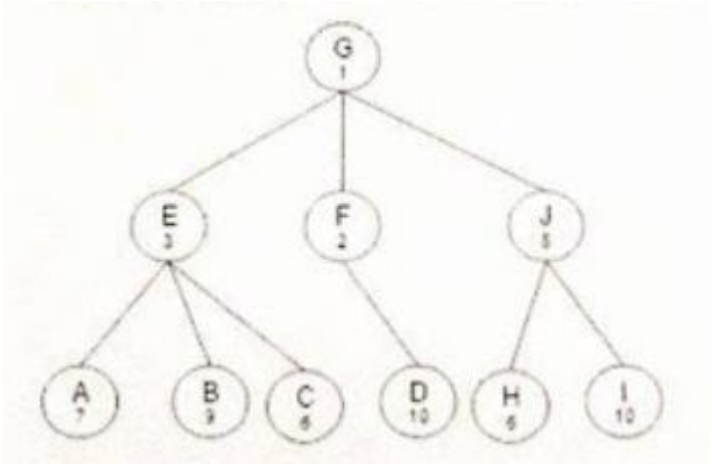
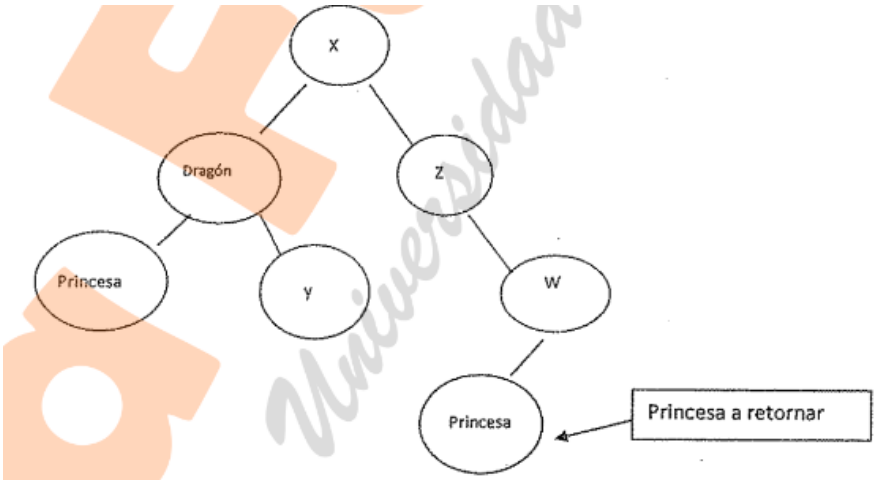
1) Sea un árbol general en donde cada nodo tiene un valor numérico, usted debe devolver el camino determinado por el valor que posee cada nodo. Es decir, el camino comienza en el nodo raíz, luego debe seguir por el hijo que se encuentra en la posición indicada por el valor en el nodo raíz y así debe seguir hasta llegar a su hoja. Para el siguiente árbol, el camino a retornar es GEC

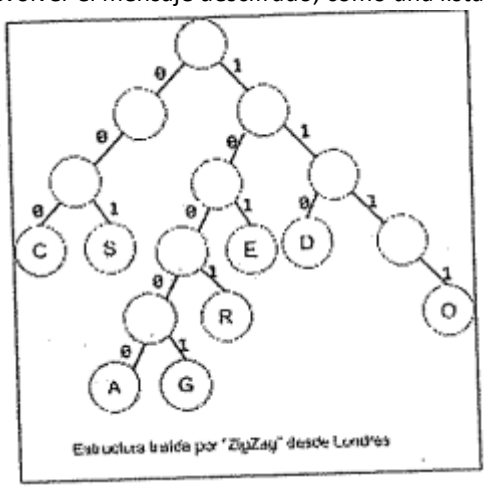


2) Las operaciones esDragon()\: boolean y esPprincesa(): boolean permiten averiguar si un persona es un dragón o una princesa respectivamente
Suponemos que ningún personaje es un dragón y princesa a la vez y que un persona puede no ser ninguna de las dos cosas. Dado un árbol binario de personajes, se denominan nodos accesibles a aquellos nodos tales que a lo largo de la raíz hasta la hoja NO se encuentra ningún dragón
Debe implementar un método princesaAccesible(): Personaje en la clase árbol binario que encunetre una princesa accesible

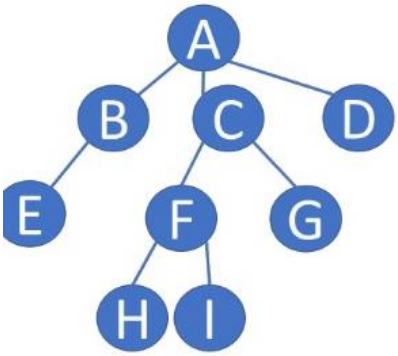


3) Llamaremos a un árbol general **creciente** si en cada nivel del árbol la cantidad de nodos que hay en ese nivel es igual al valor del nivel más 1. Es decir, el nivel 0 tiene exactamente un nodo, el nivel 1 tiene exactamente 2 nodos, el nivel k tiene exactamente k+1 nodos. Comprobar si un árbol general es creciente y en caso de que lo sea, retornar el nodo del árbol con mayor cantidad de hijos, en caso de no serlo, retornar null
4) El famoso espia Eddie Chapman, alias agente “ZigZag” está tratanto de descifrar el mensaje que le lelga desde Londres. Antes de dejar Londes, al agente le explicaron cómo debía describar el mensaje. Los mensajes estaban formados por letras, cada letra es codificada como una secuencia de unos o ceros y de diferentes longitudes. Para poder descifrar el mensaje, a ZigZag le entregaron una estructura similar a este gráfico.
En esta estructura, los 0 siempre están a la izquierda del nodo superior y los 1 a la derecha del nodo superior. A modo de ejemplo, le enviaron el siguiente mensaje cifrado: 101 001 10001 1111 1001 110 1111 con la estructura traída desde Londres, el mensaje se descibra como ESGORDO
Asuma que todas las secuencias cifradas que reicbe se corresponden con una letra. Su misión es escribir en una clase llamda CodigoZigZag un método con la siguiente firma: public ListaGeneriza<Character>

descifrarCodigo(XXXX, ListaGenerica<String> listaDeSecuencias) que sea capaz de descifrar un mensaje que ha sido sifrado como secuencias de 0 y 1. Es decir, dada la estructura traída por el agente y una lista de string usted deberá devolver el mensaje descifrado, como una lista formada por letras



5) Dado un árbol general, encontrar todos los caminos desde la raíz a una hoja, tales que la cantidad de nodos en el camino sea un número par. Escribir el siguiente método dentro de la clase Parcial **public ?? caminosPares(ArbolGeneral<Character> arbol)**



Caminos con cantidad de nodos pares:

A C F H

A C F I

A D

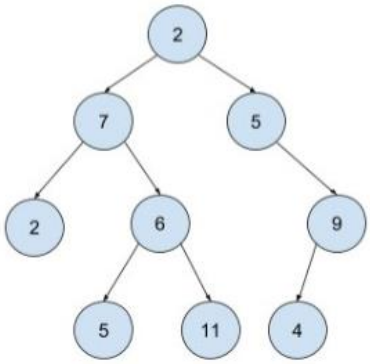
Caminos con cantidad de nodos impares:

A B E

A C G

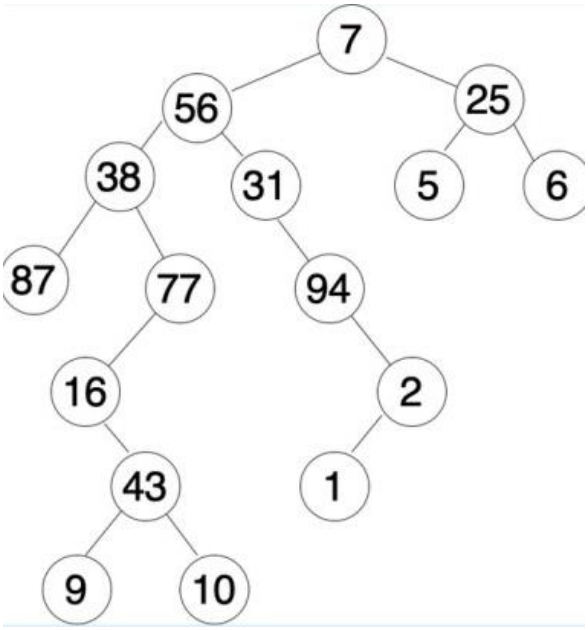
6) Sea la clase NivelArbol que tiene una variable de instancia árbol con un ArbolBinario<integer> ya inicializado. Usted debe implementar el método minEnNivelAB(int n): ArbolBinario<Integer> que devuelve el subárbol hoja con menor valor en el nivel n del árbol. De haber más de uno devuelve el primero encontrado. Considere que n es un nivel válido del árbol. Sin embargo, puede suceder que no existan hojas en ese nivel, en ese caso, debe devolver null. Realiza el recorrido por niveles
Para el siguiente árbol

- Si el nivel es 1, debe devolver null porque 7 y 5 no son hojas
- Si el nivel es 3, debe devolver 2 ya que es la única hoja
- Si el nivel es 3, debe devolver 4 porque es el menor entre 5, 11 y 4

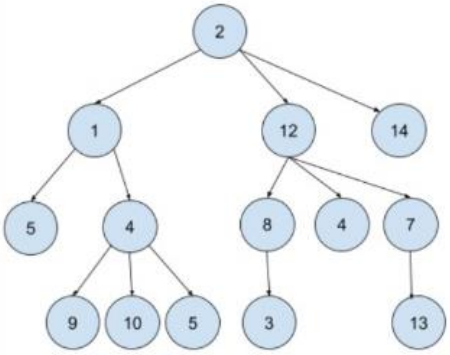


7) Implemente en la clase Parcial el método **sumaImparesPosOrdenMayorA** que recibe un árbol binario de enteros positivos y un número entero. Este método suma todos los números impares del árbol que son mayores al parámetro recibido realizandolo en un recorrido posOrder

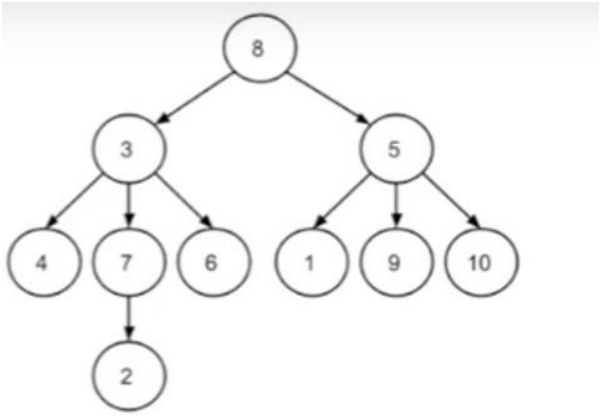
Public Integer sumaImparesPosOrdenMayorA(ArbolBinario<Integer> ab, int limite)
Por ejemplo: dado el siguiente árbol y límite siendo 30, deberá retornar 238= 87+43+77+31



8) Implemente en la clase Parcial que tiene como variable de instancia un ArbolGeneral<Integer>, el método **ListaGenerica<Integer> resolver()** que devuelva en la lista la suma de todos los datos contenidos en los nodos del árbol que tiene un número impar de hijos. Realiza un recorrido en postorden
   - en el árbol de raíz 4, devuelve 24
   - en el de raíz 8, devuelve 3
   - en el árbol de raíz 7, devuelve 13
   - en el árbol de raíz 12, devuelve 19
   - en el árbol de raíz 2, devuelve 27

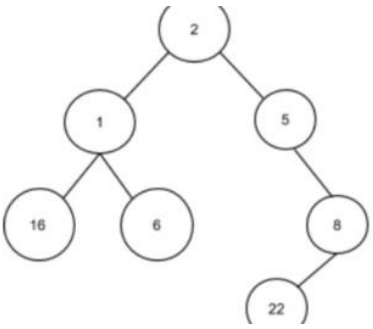


9) Devolver el camino a la hoja mas lejana. Si hubiese más de un camino de igual longitud, devolver el primero encontrado



10) Implemente en la clase Parcial el método con la siguiente firma: **resolver(ArbolBinario<Integer> arbol)** que devuelva un alista con los valores de los nodos que tengan el mismo número de descendientes tanto por su subárbol izquierdo como por su subárbol derecho.

Se debería devolver una lista con 2, 1, 16, 6, 22



11) Sea un árbol general de enteros, realizar un recorrido postorden para retornar el elemento mayor. Por ejemplo, sea el siguiente árbol, debe retornar el valor 22