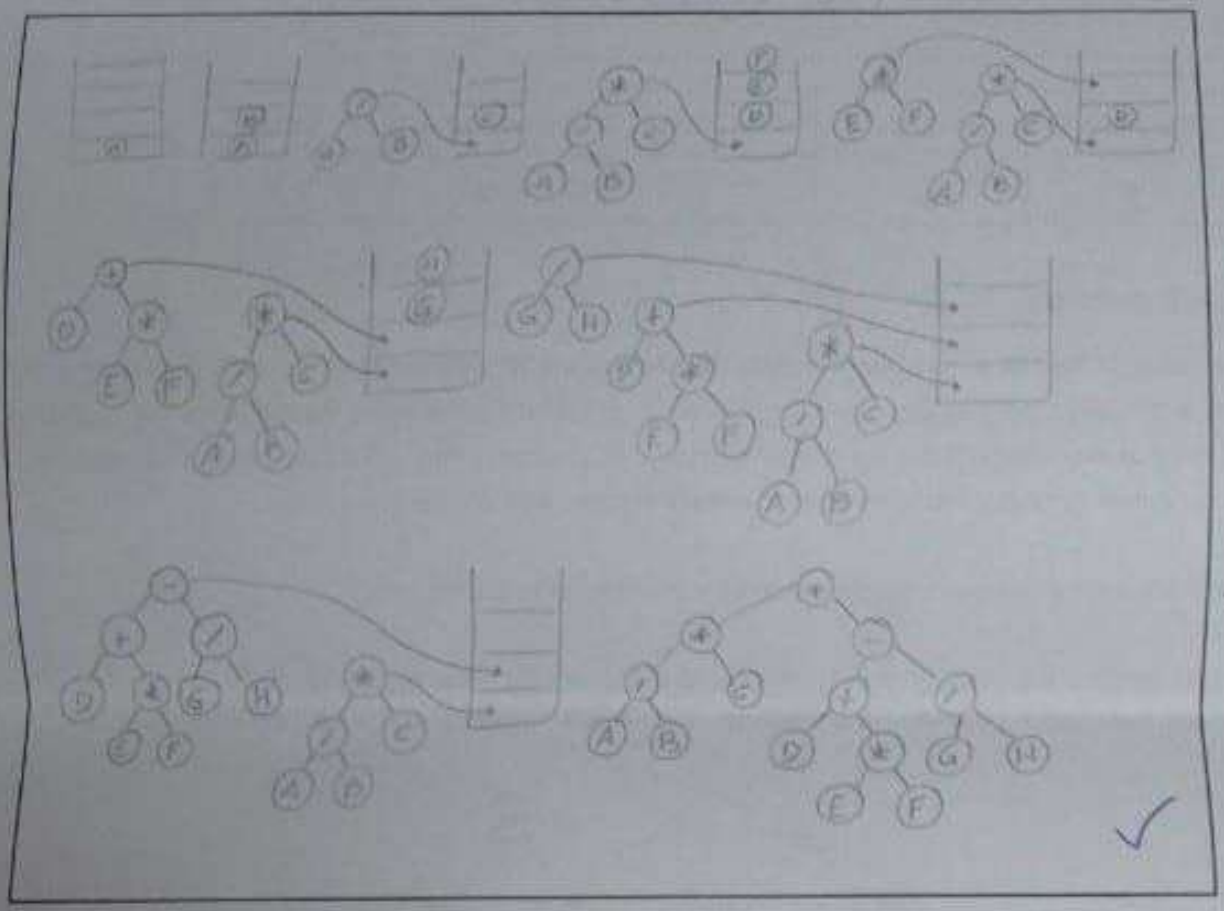
Ejercicio 2 -- 2 puntos

Construya el árbol de expresión a partir de la siguiente expresión, muestre cada uno de los pasos seguidos hasta completarlo

A B / C * D E F * + G H / - +

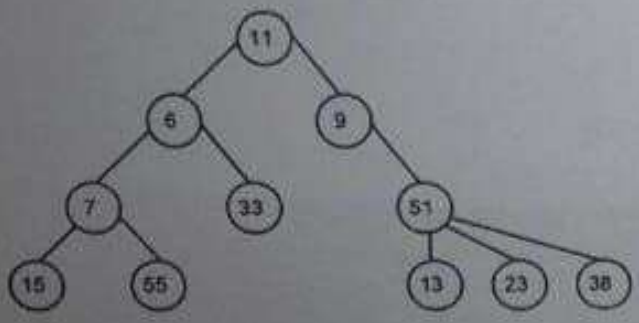


Ejercicio 3 -- 1 punto

a.- Dado un árbol general completo de grado k=4, que tiene 22 nodos en total, ¿cuál es la altura del árbol?

(a) 2 (b) 3 (c) 4 (d) 5 (e) Ninguna de las otras opciones

b. Dado el siguiente árbol general, ¿Cuál de las siguientes opciones representa el recorrido **Inorden**?



(a) 15 7 55 6 33 11 9 51 13 23 38
(b) 15 7 55 6 33 11 9 13 51 23 38
(c) 15 7 55 6 33 11 13 51 23 38 9
(d) 15 7 55 6 33 11 51 13 23 38 9

AyED Redictado 2022 - Parcial Módulo I - Tema 1

Sábado 15 de Octubre de 2022 - 9.00 horas

Apellido	Nombre	Legajo	Turno	Corrigió
[redacted]				

Ejercicio 1	Ejercicio 2	Ejercicio 3	Ejercicio 4	Total
5	2	1	2	10

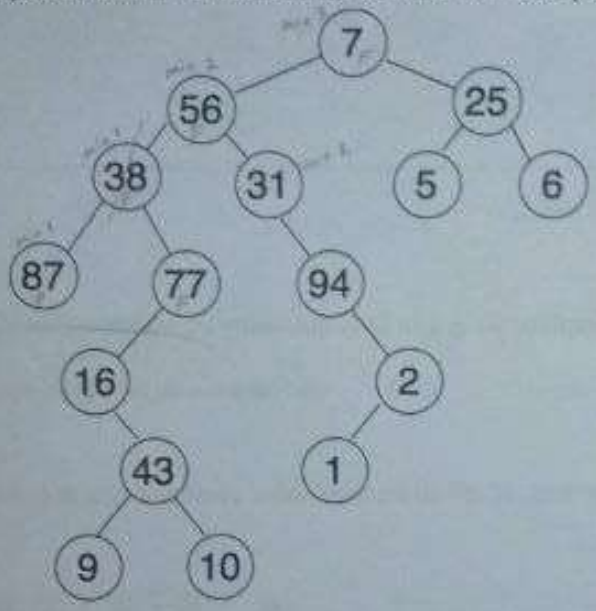
Ejercicio 1 -- 5 puntos

Implemente en la clase **Parcial** el método **resolver** que recibe un árbol binario de enteros positivos y un número entero y devuelve un camino que cumple con la siguiente condición: la cantidad de números pares que contenga dicho camino debe ser mayor o igual al parámetro "min". Si existen varios caminos que cumplen la condición, el método debe devolver el primer camino que encuentre.

```
public ListaGenerica<Integer> resolver(ArbolBinario<Integer> ab, int min)
```

Por ej. dado el siguiente árbol y siendo min= 2 retornará el camino formado por: 7-56-38-87

Por ej. dado el siguiente árbol y siendo min=3 retornará el camino formado por: 7-56-38-77-16-43-9



Tenga en cuenta que:

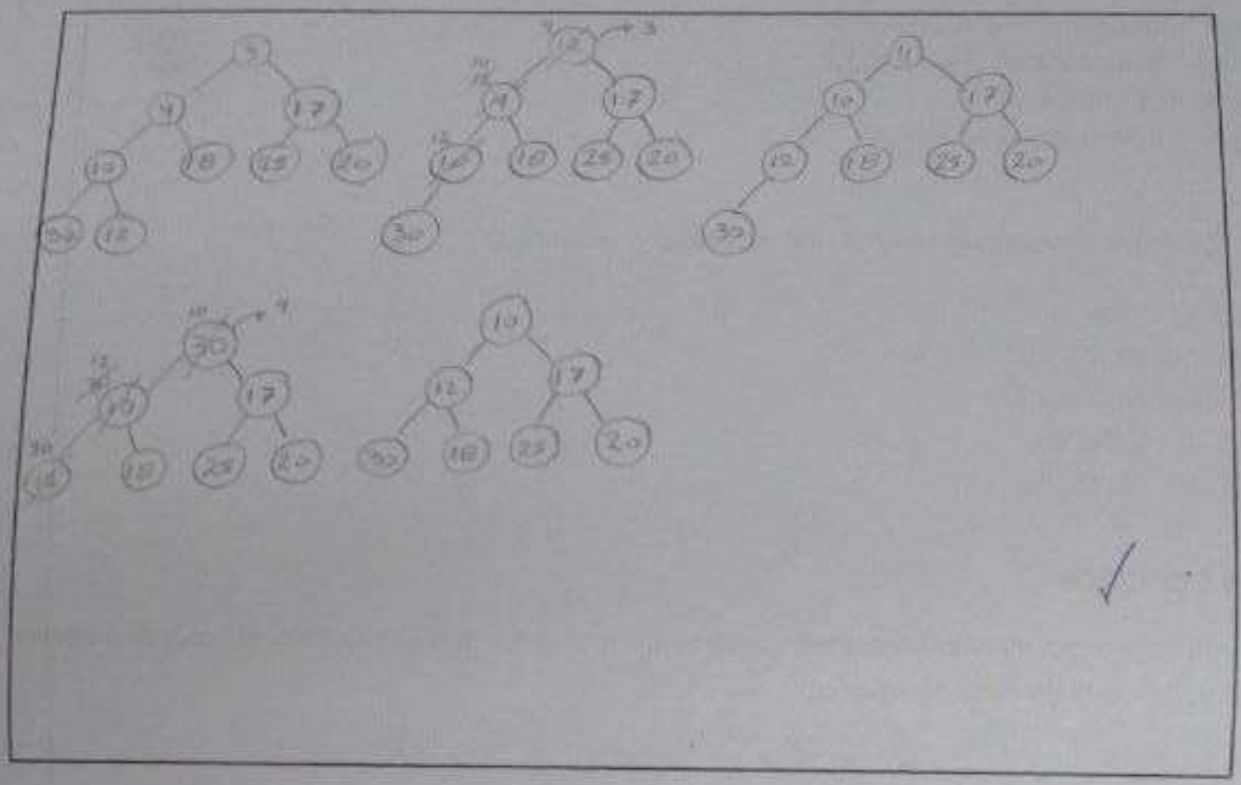
- Debe recorrer la estructura solo 1 vez para resolverlo.
- Debe respetar la clase y el método indicado.
- Puede definir todos los métodos y variables auxiliares que considere necesarios.
- Todo método que no esté definido en las sinopsis de clases debe ser implementado.

-- 2 puntc

rbol de expr

ario

b.- Realice dos operaciones de DeleteMin()



✓

-- 1

bol g

uient

Ejercicio 2

Construya el á
hasta complet

c. Un árbol binario COMPLETO de altura h, h ≥ 0 tiene:

(a) Exactamente 1 nodo hoja en el nivel h
((b)) Como mínimo 1 nodo hoja en el nivel h
(c) Ninguna de las otras opciones
(d) Como máximo 1 nodo hoja en el nivel h

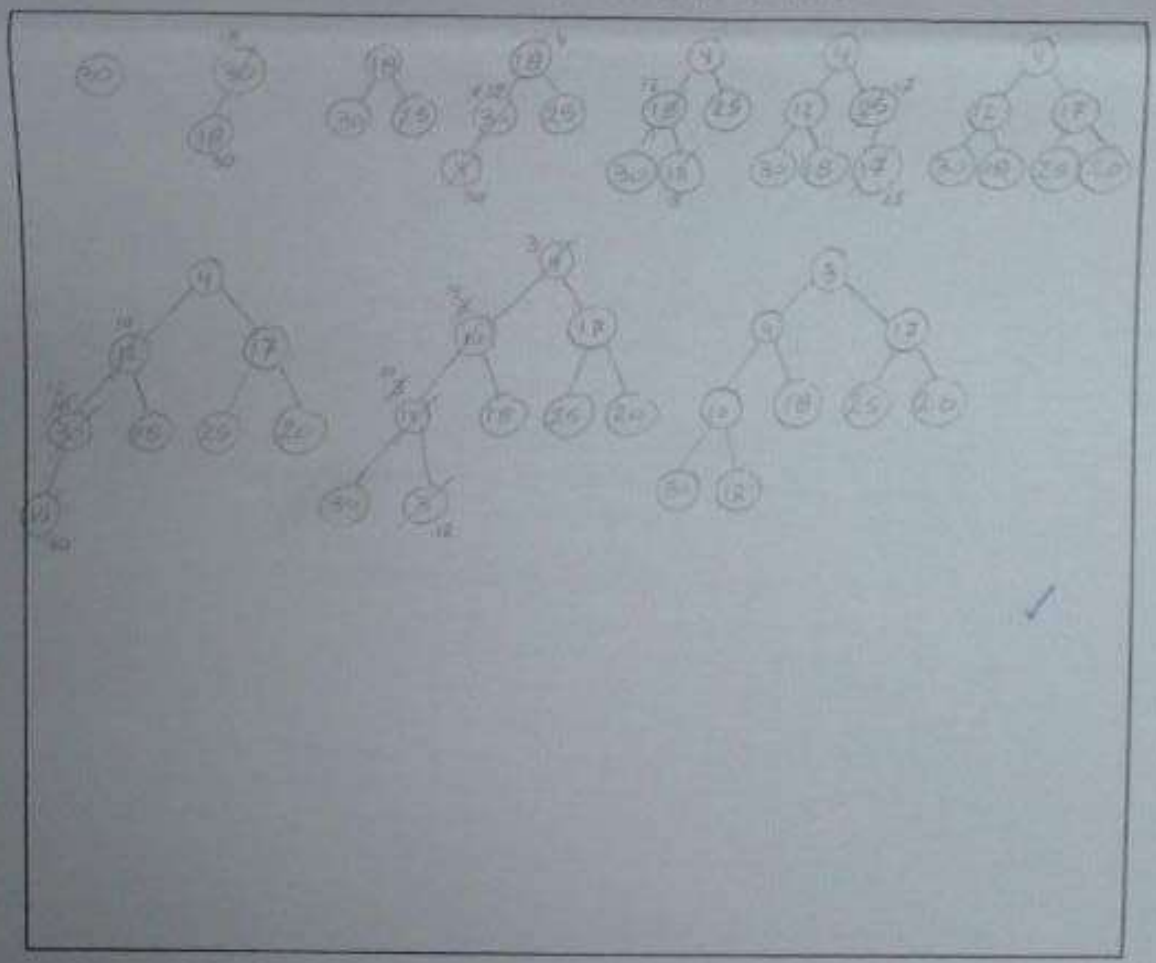
d. ¿Cuál de los siguientes arreglos representa una max-heap o min-heap?

(a) 50, 25, 13, 6, 18, 24, 40
(b) 13, 30, 25, 50, 38, 27, 40
(c) 13, 25, 30, 50, 38, 17, 40
(d) 50, 25, 13, 18, 6, 24, 40
(e) 13, 30, 17, 50, 38, 15, 40

Ejercicio 4 -- 2 puntos

a.- A partir de una min-heap inicialmente vacía, inserte de a uno los siguientes valores, muestre cómo evoluciona la heap, después de cada operación:

30, 18, 25, 4, 12, 17, 20, 10, 3



[redacted]

```
public class Pascal {

  public ListaGenerica<Integer> resolver(ArbolBinario<Integer> ab, int num) {
    ListaGenerica<Integer> lista = new ListaEnlazadaGenerica<Integer>();

    if(!ab.esVacio())
      resolver(ab, num, lista);

    return lista;
  }

  private void resolver(ArbolBinario<Integer> ab, int num, ListaGenerica<Integer> lista) {
    boolean encontre = false;
    int aguja = ab.getDato();
    if(aguja == num)
      encontre = true;

    if(ab.tieneHijoIzquierdo()) {
      if(!encontre) {
        encontre = true;
        lista.agregarFinal(aguja);
      } else
        lista.agregarFinal(ab.getDato());
    }

    if(ab.tieneHijoDerecho()) {
      if(!encontre)
        resolver(ab.getHijoDerecho(), num, lista);

      if(ab.tieneHijoIzquierdo() && encontre)
        resolver(ab.getHijoDerecho(), num, lista);

      if(encontre)
        lista.agregarFinal(ab.getDato());
    }

    return encontre;
  }
}
```

```
public class Pacel {

public ListaGenerica<Integer> resolver (ArbolBinario<Integer> ab, int min) {
  ListaGenerica<Integer> lista = new ListaEnlazadaGenerica<Integer>();

  If(!ab.esVacio())
    resolver2 (ab, min, lista);

  return lista;

}

private boolean resolver2 (ArbolBinario<Integer> ab, int min, ListaGenerica<Integer> lista) {
  boolean encontre = false;
  lista.agregarFinal(ab.getDato());
  If (ab.getDato % 2 == 0)
    min = min - 1;

  If(ab.esHoja()) {
    If (min <= 0) {
      encontré = true;
      return encontre;
    } else
      lista.eliminar(lista.tamanio());

  } else {
    If (ab.tieneHijoIzquierdo() && !encontre)
  encontre = resolver2 (ab.getHijoIzquierdo(), min, lista);


    If (ab.tieneHijoDerecho() && !encontre)
  encontre = resolver2 (ab.getHijoDerecho(), min, lista);

    If (!encontre)
      lista.eliminar(lista.tamanio());


  }
  return encontre;
}

}
```