```java
//Classe ArvoreBinaria;
//Questão A == preFixEsq();
//Questão B == preFix2();
import java.util.ArrayDeque; import java.util.Deque;
public class ArvoreBinaria {
    private No raiz;
    public static int qtdeNoEsquerda = 0;
    public ArvoreBinaria() {
        this.raiz = null;
    }
    public No getArvore() {
        return raiz;
    }
    public void setArvore(No raiz) {
        this.raiz = raiz;
    }
    public void insere(long id, Object elemento) {
        No novoNo = new No(id, elemento, null, null);
        if (raiz == null) {
            raiz = novoNo;
        } else {
            No atual = raiz;
            No pai;
            while (true) {
                pai = atual;
                if (id < atual.getId()) {
                    atual = atual.getEsq();
                    if (atual == null) {
                        pai.setEsq(novoNo);
                        return;
                    }
                } else {
                    atual = atual.getDir();
                    if (atual == null) {
                        pai.setDir(novoNo);
                        return;
                    }
                }
            }
        }
    }
    public int preFixEsq(No atual) {
        try {
            if (atual.getEsq() != null) {
                qtdeNoEsquerda++;
            }
        } catch (Exception e) { }
        if (atual != null) {
            preFixEsq(atual.getEsq());
```

```
            preFixEsq(atual.getDir());
        }
        return qtdeNoEsquerda;
    }
    public void preFix2(No atual, int qnt) {
        String espaço = "";
        for (int i = 1; i <= qnt; i++) {
            espaço += "   ";
        }
        if (atual != null) {
            qnt++;
            System.out.println(espaço + atual.getId());
            if (atual.getEsq() == null) {
                System.out.println(espaço + "    -");
            }
            if (atual.getDir() == null) {
                System.out.println(espaço + "    -");
            }
            preFix2(atual.getEsq(),  qnt);
            preFix2(atual.getDir(), qnt );
        }
    }
    public void imprimeEleArvore() {
        preFixEsq(raiz);
    }
    public void imprimeEleArvore2() {
        preFix2(raiz, 0);
    }
    public void imprimeArvore() {
        percorreArvore(raiz);
    }
    private long calcAltura(No atual, long a) {
        if (atual != null) {
            long e, d;
            e = calcAltura(atual.getEsq(), a) + 1;
            d = calcAltura(atual.getDir(), a) + 1;
            if (e > d) {
                return a + e;
            } else {
                return a + d;
            }
        }
        return a;
    }
    public long alturaArvore() {
        long a = 0;
        return calcAltura(raiz, a);
    }
    public String PreOrdem() {
```

```
        if (getArvore() == null) {
            return "";
        }
        return readPreOrdem(getArvore());
    }
    private String readPreOrdem(No no) {
        if (no == null)  {
            return "";
        }
        String rt = no.toString();
        if (no.getEsq() != null) {
            rt += (rt.isEmpty() ? "" : ",") + readPreOrdem(no.getEsq());
            qtdeNoEsquerda++;
        }
        if (no.getEsq() != null) {
            rt += (rt.isEmpty() ? "" : ",") + readPreOrdem(no.getDir());
        }
        return rt;
    }
    private long calcAlturaNoEsq(No atual, long a) {
        if (atual != null) {
            long e = calcAltura(atual.getEsq(), a) + 1;
            return e;
        }
        return a;
    }
    public long alturaArvoreNoEsq() {
        long a = 0;
        return calcAlturaNoEsq(raiz, a);
    }
    public void percorreArvore(No no) {
        int qtNoEsq = 0, qtNoDir = 0;
        Deque<No> fila = new ArrayDeque<>( );
        fila.add(no);
        while (!fila.isEmpty()) {
            No atual = fila.removeFirst();
            System.out.printf("%s, ", atual.getElemento());
            if (atual.getEsq() != null) {
                fila.add(atual.getEsq());
                qtNoEsq++;
            }
            if (atual.getDir() != null) {
                fila.add(atual.getDir());
                qtNoDir++;
            }
        }
        System.out.println("qtNoEsq= " + qtNoEsq);
    }
}
```

```java
//Classe No;
public class No {
    private long id;
    private Object elemento;
    private No esq,dir;
    public No(long id, Object elemento, No esq, No dir){
        this.id = id; this.elemento = elemento; this.esq = esq; this.dir =
dir;
    }
    public void setId(long id) { this.id = id; }
    public long getId() { return this.id; }
    public void setElemento(Object elemento) { this.elemento = elemento; }
    public Object getElemento() { return elemento; }
    public void setEsq(No esq) { this.esq = esq; }
    public No getEsq() { return esq; }
    public void setDir(No dir) { this.dir = dir; }
    public No getDir() { return dir; }
    public String toString() { return getElemento() == null ? "" : (String)
getElemento(); }
}
```

```java
//Classe ExemploDeArvoreBinaria para validar os algorítmos;
public class ExemploDeArvoreBinaria {
    public static void main(String[] args) {
        System.out.println("\nQuestão A");
        System.out.println("\n");
        ArvoreBinaria treeone = new ArvoreBinaria();
        treeone.insere(10,"A");
        treeone.insere(5,"B");
        treeone.insere(15," C");
        treeone.insere(12," D");
        treeone.insere(7,"E");
        treeone.insere(12,"F");
        treeone.insere(6,"G");
        treeone.insere(8,"H");
        treeone.insere(17,"I");
        treeone.insere(11,"J ");
        treeone.insere(14,"K");
        treeone.insere(3,"L");
        treeone.imprimeEleArvore();
        System.out.println("Quantidade de Nó a Esquerda: "  +
treeone.qtdeNoEsquerda);
        System.out.println("\nAltura: " + treeone.alturaArvore());
        System.out.println("\nPreOrdem: " + treeone.PreOrdem());
        System.out.println("\n");
        System.out.println("\n");
        System.out.println("Questão B");
        System.out.println("\n");
```

```
        String espaço;
        ArvoreBinaria treetwo = new ArvoreBinaria();
        treetwo.insere(555, "A");
        treetwo.insere(333, "B");
        treetwo.insere(888, "C");
        treetwo.insere(111, "D");
        treetwo.insere(444, "E");
        treetwo.insere(999, "F");
        treetwo.imprimeEleArvore2();
    }
}
```