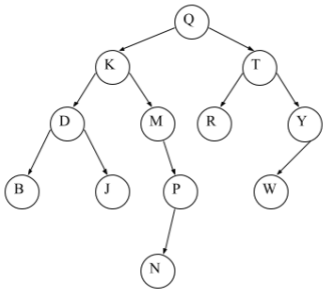
BST y ABB

Nombre: Emilio Fernando Alonso Villa
Matricula: A00959385

I. Dada el BST que a continuación se presenta, contesta las siguentes preguntas.



1) ¿En qué nivel está el nodo con información "M"?
Respuesta: En el nivel 2

2) ¿Cuál es la altura del árbol?
Respuesta: 5

3) ¿Cuáles son los acenstros de "Y"?
Respuesta: T, Q

4) ¿Cómo son visitados los nodos por el recorrido en Preorden?
Respuesta: Q, K, D, B, J, M, P, N, T, R, Y, W

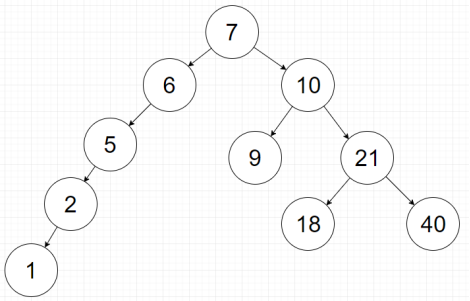
5) ¿Cómo son visitados los nodos por el recorrido Inorden?
Respuesta: B, D, J, K, M, N, P, Q, R, T, W, Y

6) ¿Cómo son visitados los nodos por el recorrido en Postorden?
Respuesta: B, J, D, N, P, M, K, R, W, Y, T, Q

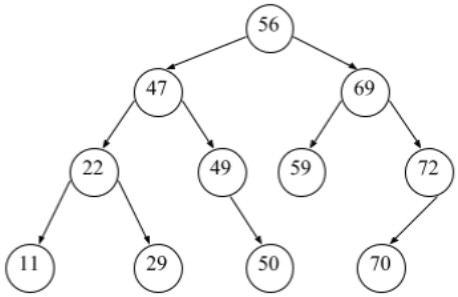
II. Para la siguiente secuencia de datos (llegan de izquierda a derecha), genera el BST.

7, 10, 21, 40, 6, 18, 9, 5, 2, 1

RESPUESTA



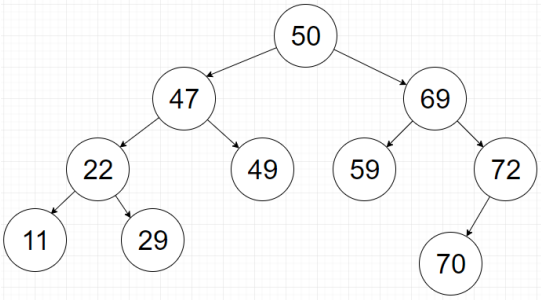
III. Para el siguiente árbol binario



Contesta las siguientes preguntas, asumiendo que el árbol en su estado inicial para cada una de ellas.

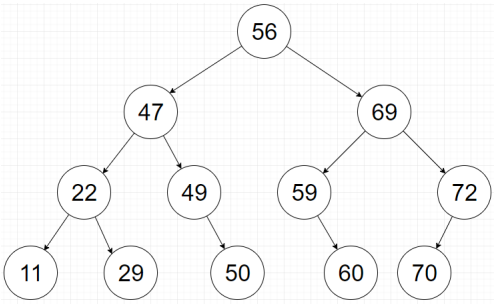
1. Asumiendo que el árbol es un ABB ¿Cómo quedaría el ABB si se borra el 56 usando el predecesor?

RESPUESTA



2. Asumiendo que el árbol es un ABB ¿Cómo quedaría el ABB si se inserta el 60?

RESPUESTA



IV. Implementa las siguientes operaciones del BST.

printLeaves	
Descripción:	Despliega el contenido de las hojas del árbol
Entrada:	Ninguna
Salida:	Ninguna
Pre-condición:	Que el árbol exista
Post-condición:	Nada

RESPUESTA

```
#include <queue>
void BST::printLeaves(){
    // checa que el árbol exista
    if(!root){
        return;
    }

    else{
        queue<NodeT *> leafQueue;
        leafQueue.push(root);

        while(!leafQueue.empty()){
            NodeT *temp = leafQueue.front();
            leafQueue.pop();

            if(temp->getLeft()){
                leafQueue.push(temp->getLeft());
            }
            if(temp->getRight()){
                leafQueue.push(temp->getRight());
            }
            // Sólo imprime si no hay nodos ni en izq ni en derecha
            if(!temp->getLeft() && !temp->getRight()){
                cout << temp->getData() << endl;
            }
        }
    }
}
```

count	
Descripción:	Regresa la cantidad de nodos en el árbol
Entrada:	Ninguna
Salida:	Un entero, que significa la cantidad de datos del árbol
Pre-condición:	Que el árbol exista

count	
Post-condición:	Nada

RESPUESTA

```
#include <queue>
int BST::count(){
    // Checa que el árbol exista
    if(root == nullptr){
        return 0;
    }

    else{
        queue<NodeT *> nodeQueue;
        int nodeCount = 1; // inicia en 1 por la raiz
        nodeQueue.push(root);

        while(!nodeQueue.empty()){
            NodeT *temp = nodeQueue.front();
            nodeQueue.pop();

            if(temp->getLeft()){
                nodeQueue.push(temp->getLeft());
                nodeCount++;
            }
            if(temp->getRight()){
                nodeQueue.push(temp->getRight());
                nodeCount++;
            }
        }
        return nodeCount;
    }
}
```