In [8]:

```
class Nodo:
    def __init__(self, dato):
        self.dato = dato
        self.izquierda = None
        self.derecha = None

class Arbol:
    def __init__(self, dato):
        self.raiz = Nodo(dato)

    def __agregar_recursivo(self, nodo, dato):
        if dato < nodo.dato:
            if nodo.izquierda is None:
                nodo.izquierda = Nodo(dato)
            else:
                self.__agregar_recursivo(nodo.izquierda, dato)
        else:
            if nodo.derecha is None:
                nodo.derecha = Nodo(dato)
            else:
                self.__agregar_recursivo(nodo.derecha, dato)

    def __inorden_recursivo(self, nodo):
        if nodo is not None:
            self.__inorden_recursivo(nodo.izquierda)
            print(nodo.dato, end=", ")
            self.__inorden_recursivo(nodo.derecha)

    def __preorden_recursivo(self, nodo):
        if nodo is not None:
            print(nodo.dato, end=", ")
            self.__preorden_recursivo(nodo.izquierda)
            self.__preorden_recursivo(nodo.derecha)

    def __postorden_recursivo(self, nodo):
        if nodo is not None:
            self.__postorden_recursivo(nodo.izquierda)
            self.__postorden_recursivo(nodo.derecha)
            print(nodo.dato, end=", ")

    def __buscar(self, nodo, busqueda):
        if nodo is None:
            return None
        if nodo.dato == busqueda:
            return nodo
        if busqueda < nodo.dato:
            return self.__buscar(nodo.izquierda, busqueda)
        else:
            return self.__buscar(nodo.derecha, busqueda)


    def agregar(self, dato):
        self.__agregar_recursivo(self.raiz, dato)

    def inorden(self):
        print("Imprimiendo árbol inorden: ")
        self.__inorden_recursivo(self.raiz)
        print("")
```

```
    def preorden(self):
        print("Imprimiendo árbol preorden: ")
        self.__preorden_recursivo(self.raiz)
        print("")

    def postorden(self):
        print("Imprimiendo árbol postorden: ")
        self.__postorden_recursivo(self.raiz)
        print("")

#Se procede a cargar el Arbol
arbol = Arbol("8")
arbol.agregar("9")
arbol.agregar("11")
arbol.agregar("15")
arbol.agregar("19")
arbol.agregar("20")
arbol.agregar("21")
arbol.agregar("7")
arbol.agregar("3")
arbol.agregar("2")
arbol.agregar("1")
arbol.agregar("5")
arbol.agregar("6")
arbol.agregar("4")
arbol.agregar("13")
arbol.agregar("14")
arbol.agregar("10")
arbol.agregar("12")
arbol.agregar("17")
arbol.agregar("16")
arbol.agregar("18")

print ("Modulo 3 \nUrbanoIguala\nalinsertar el arbol binario y recorrerlo en Preorden, In
print ("Podemos observr que n el recorrido InOrden, los nodos tienen una secuencia toman
print ("el primer valor para determinar su secuencia")
arbol.preorden()
arbol.inorden()
arbol.postorden()
```

```
Modulo 3
UrbanoIguala
alinsertar el arbol binario y recorrerlo en Preorden, InOrden y PostOrden
Podemos observr que n el recorrido InOrden, los nodos tienen una secuencia tomando en cue
nta unicamente
el primer valor para determinar su secuencia
Imprimiendo árbol preorden:
8, 11, 1, 10, 15, 13, 12, 14, 19, 17, 16, 18, 20, 2, 21, 7, 3, 5, 4, 6, 9,
Imprimiendo árbol inorden:
1, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 2, 20, 21, 3, 4, 5, 6, 7, 8, 9,
Imprimiendo árbol postorden:
10, 1, 12, 14, 13, 16, 18, 17, 2, 4, 6, 5, 3, 7, 21, 20, 19, 15, 11, 9, 8,
```

In [ ]:

In [ ]: