Modelo de Parcial 2

Ejercicio 1: Considere trabajar con el TAD **árbol binario de búsqueda**.

a) Defina el objeto de datos.

b) Especifique e implemente la operación que cuente la **cantidad de nodos que tienen un solo descendiente directo.**

```python
# a) Defina el objeto de datos.
class Nodo:
    __der:object
    __izq:object
    __elem: str

    def __init__(self,x):
        self.__der = None
        self.__izq = None
        self.__elem = x

class Arbol:
    __raiz:Nodo

    def __init__(self):
        self.__raiz = None

def grado(self):
    g = 0
    if self.__izq is not None:
        g += 1
    if self.__der is not None:
        g += 1
    return g

# b) Especifique e implemente la operación  que cuente la cantidad de nodos que tienen
# un solo descendiente directo.

# Especificación
# Nombre: PreOrden
# Funcion: Procesa A en Preorden
# Salida: Sujeta al proceso que se realice sobre los elementos de A


    def preorden(self,nodo):
        c = 0

        if nodo != None:
            if nodo.grado() == 1:
                c += 1
                print(f"Nodos: {nodo.getDato()}")
            c += self.preorden(nodo.getIzq())
            c += self.preorden(nodo.getDer())

        return c
```

Ejercicio 3: : Considere el TAD **digrafo**, formado por N vértices.

a) Defina el objeto de datos.

Especifique e implemente la operación que **muestre todos los nodos sumidero, para ello implemente las operaciones grado de entrada y grado de salida**

```python
# a) Defina el objeto de datos.
class Digrafo:
    __vertices: int
    __matriz: np.ndarray

    def __init__(self,N):
        self.__matriz = np.zeros((N,N),dtype=int)
        self.__vertices = N

    def grado_entrada(self, u):
        cont = 0

        for i in range(self.__vertices):
            if self.__matriz[i][u] == 1:
                cont += 1

        return cont

    def grado_salida(self, u):
        return len(self.adyacentes(u))

    # Operación: Nodo Sumidero
    # Funcion: Evalúa si u es nodo sumidero de G
    # Salida: V si u es nodo sumidero de G

    def nodoSumideros(self):
        band = False

        for u in range(self.__vertices):
            if self.grado_salida(u) == 0 and self.grado_entrada(u) > 0:
                print(f"Nodos Sumideros: {u}")
                band= True

        if not band:
            print(f"No existen nodos sumideros")
```