Tecnología de la Programación **Extraordinaria de 2024**
Duración: 3h:15m Evaluación Final.

Las soluciones tienen que estar razonadas. Comenta el código. Justifica el valor de retorno de los métodos. Se asume que todos los atributos privados tienen propiedades get/set (no los escribas).
ENTREGA EL ENUNCIADO JUNTO CON TUS SOLUCIONES.

Apellidos y Nombre: .....

Ejercicio 1 (2 puntos + (1 Extra; si llega al menos a un 6))

(A) En la clase `Grafo` construye un método **no recursivo** que retorne la lista de vértices que resultante de realizar un recorrido en profundidad a partir de un vértice dado y que cumplan cierta función `test(valor: int)->bool`.

- Cada vértice del grafo será un número natural y todos son distintos.
- Recuerda que el recorrido no puede contener al nodo más de una vez.
- El grafo se codifica con un lista de adyacencia usando un diccionario: la clave es un vértice (int) y el valor es la lista de vértices adyacentes (List[int]).
- **Solo** se considera solución válida la lista de vértices (que son enteros) que cumplan la función test() y que se obtiene **directamente** del recorrido en profundidad. **No** será solución válida la que obtenga la lista de vértices en profundidad y después filtre la lista con la función `test()`.
- ¿Cómo se definirá la función `test()` para que el resultado sea el recorrido en profundidad completo del grafo?

(B) (+1 Extra) Construye un iterador para recorrer los nodos de un grafo usando un recorrido en profundidad.

Ejercicio 2 (6 puntos = 1+2+1+2)

[*Son **visitantes** (de una feria), los **menores** de edad y los **adultos**. De todo visitante se conoce su edad (no negativa) **y** si tiene derecho a un descuento. El menor de edad tiene menos de 18 años y el adulto tiene 18 años o más. Si un menor de edad tiene 8 años o menos (es infantil) o si un adulto tiene 65 años o más (es mayor) tienen derecho a un descuento.*] **1** Implementa las clases con los atributos y métodos mencionados comprobando el invariante de la representación en el constructor he indicando, usando la notación de Python, sus modificadores de visibilidad y si tendría una propiedad o método set y/o get en forma de comentario. P.e. `self._salero: Set[Sal] # get, set` representa un atributo protegido y su valor es un conjunto cuyos elementos son granos de sal y tiene sentido hacer un get y set sobre dicho atributo. `self.__notas: Dict[DNI, int] # get` representa un diccionario privado al que solo se puede consultar la calificación numérica asociada a un DNI.

[*Una cola de visitantes, `ColaVisitantes`, es una clase hija del TDA Cola y cuyos elementos son objetos de tipo visitante. Dispone de un método que permite guardar los visitantes de la cola en un fichero de texto que genera un **error propio** si los datos no pueden ser guardados.*] **2** ¿Cómo se construye e invoca a dicho método teniendo en cuenta que puede generar un error propio?

```
for clase in [MenorEdad, Adulto]:
    ...
    edad=randint(0, 100)
    visitante = clase(edad)
    ...
```

[*`ColaVisitantes` dispone del método factoría `def cola_aleatoria(n: int)->ColaVisitantes` que permite la construcción de una cola de n-visitantes aleatorios, de cada tipo, y que incluye el código de la izquierda.*] **3** Complete el código, respetando el que ya aparece, para que se genere una cola de 10 visitantes de cada clase, usando las clases de 1, y añáde el método factoría a la clase `ColaVisitantes` del apartado 2. ¿Cómo se invoca al método?

[*Tanto el recito ferial (solo existirá uno) como cada atracción (hay varias) tienen un aforo (son valores sin relación entre ellos). Tanto en el recinto como en las atracciones, los visitantes **entran** de uno en uno (e.d. hay un. método para entrar). El recinto y las atracciones conocen a todos los que entran en sus instalaciones. Cuando se alcanza un aforo no podrán entrar más visitantes (generando el correspondiente error). Además, en el caso de una atracción, solo pueden entrar los visitantes para los que está destinada la atracción, que puede ser para público infantil (< 8 años), adulto (> 18 años) o familiar (todas las edades).*]

4 Dada una cola de visitantes ¿cómo entran en el ferial y en las atracciones? Construye y modifica todas las clases que consideres necesarias, así como la función principal.

Ejercicio 3 (2 puntos) Una lista multinivel es una lista de nodos simplemente enlazada (sin cabecera) y ordenada creciente, donde cada nodo de la lista consta de un valor y de una lista de valores. Se construyen con valores numéricos o strings.
Ejemplo: La lista multinivel para los datos {10, 7, 25, 13, 6, 20, 68} es ['1' [10, 13], '2' [25, 20], '6' [6, 68], '7' [7]]. La lista enlazada está formada por los valores ordenados ['1', '2', '6', '7'] que están formados por el primer carácter del dato de entrada, empezando por la izquierda. Entonces, la sublista del '1' es [10, 13] pues el 10 y el 13 empiezan por 1, la del '2' es [25, 20], etc. Es decir cada dato que vaya a ser añadido se colocará en la sublista del nodo cuyo valor sea `str(dato)[0]`.

El objetivo del ejercicio es construir el método *.append()* del siguiente código:

```
class Nodo:
    def __init__(self, dato: Any):
        self.__dato = dato  # get, set
        self.__siguiente: Union['Nodo', None] = None  # get,
        ↪ set
        self.__sublista: List[Any] = []  # get, set

class ListaMultiNivel:
    def __init__(self):
        self.__primero: Nodo = None  # get, set. Sin cabecera.

    def append(self, dato: Any) -> None:
        dato_str = str(dato)[0] # Obtiene el primer carácter del
        ↪ dato
        ... Se pide este método ...
```

Restricciones:

1. Para la creación de la lista enlazada no se puede recurrir a ningún otro método de TDA Lista.
2. Para la sublista, solo se pude usar el método .append() de las listas de Python.
3. El úso de índices se limita solo a la expresión `str(dato)[0]`. No puede usar ningún otro índice.

Pistas: Resuelve el ejercicio en dos etapas:

1. Construir una lista de nodos simplemente enlazada, cuyos elementos no se repiten y estén ordenados de forma creciente.
2. Completar el código anterior para añadir los elementos en las sublistas