Estructuras de Datos

TP 2: TAD – Implementación

1. Implemente la pila y la cola que especificó en el Práctico Uno. En ambos casos justifique la elección de la estructura interna e indique el Orden de cada operación implementada.

2. ¿Las operaciones `length()` implementadas en el ejercicio anterior tienen O(1)? En caso de no tener O(1) indique que debería modificar tanto para la pila como para la cola para obtener Orden constante.

3. Escriba un programa que lea 10 números y los imprima en orden inverso. ¿Utilizó en la solución alguno de los TAD's que implementó en esta práctica? Si tuviera que imprimir ahora los números en el orden que fueron leídos, ¿qué TAD de los que ya implementó usaría?

4. Escriba un programa que lea una secuencia de caracteres e informe si la misma es palíndromo. Utilice en la solución los TADś implementados.

5. Escriba un programa que determine si una secuencia de llaves, corchetes y paréntesis esta balanceada. Es decir por cada símbolo de apertura debe aparecer su correspondiente símbolo de cierre. Tenga en cuanta que los símbolos pueden aparecer anidados.

   Por ejemplo la secuencia **({[()]})** es válida mientra que **[({)}]** no es válida.

6. Implemente el TAD fecha especificado en el Práctico Uno. ¿La estructura interna debería poder mutar? Ayuda: Para manejar los topes de días en los meses puede tener una lista como la siguiente:

   ```
   meses = [None, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
   ```

   Debería ver como manejar que febrero tiene 29 días si el año es bisiesto. Un año es bisiesto si es divisible entre 4, **salvo que sea año secular** (último de cada siglo, terminado en "00"), en cuyo caso debe ser divisible entre 400.

7. Implemente una cola doblemente terminada usando nodos. Las operaciones `encolar()` y `desencolar()` deben tener O(1)

8. Implemente una lista enlazada simple (como las que vimos en PI). Tenga en cuenta que debe implementar las operaciones "insertar después de..." y "borrar después de..." (a raíz de las limitaciones que esta estructura posee). Para indicar la posición donde operar deberá emplear una coordenada. Usé la siguiente estructura interna con un "nodo escondido" para implementar la lista:

   ```
   from dataclasses import dataclass
   from typing import Any

   class SinglyLinkedList():
       @dataclass
       class _Head:
           next: '_Node' = None

       @dataclass
       class _Node:
           value: Any
           next: '_Node' = None
   ```

```
    __slots__ = ['_head']

    def __init__(self, iterable=None):
        self._head = SinglyLinkedList._Head(None)
        if iterable is not None:
            prev = self._head
            for value in iterable:
                node = SinglyLinkedList._Node(value, None)
                prev.next = node
                prev = node
```

9. Implemente “Cuenta Regresiva” que se comporta como “Range” pero contando al revés.
10. ¿Qué debería agregar a su implementación para que “Cuenta Regresiva” soporte ser iterado con un “for de Python”?