Ejercitación TAD Lista

Ejercicio 1:

A partir de una estructura **LinkedList** definida de la siguiente manera:

```
class LinkedList:
    head=None
```

Y la estructura **Node** definida de la siguiente manera:

```
class Node:
    value=None
    nextNode=None
```

Crear un modulo de nombre **`linkedlist.py`** que **implemente** las siguientes especificaciones de las operaciones elementales para el **TAD secuencia** utilizando el TAD lista.

add(L,element)

Descripción: Agrega un elemento al comienzo de L, siendo L una LinkedList que representa el **TAD secuencia.**
Entrada: La Lista sobre la cual se quiere agregar el elemento (LinkedList) y el valor del elemento (element) a agregar.
Salida: No hay salida definida

search(L,element)

Descripción: Busca un elemento de la lista que representa el **TAD secuencia.**
Entrada: la lista sobre el cual se quiere realizar la búsqueda (Linkedlist) y el valor del elemento (element) a buscar.
Salida: Devuelve la posición donde se encuentra la primera instancia del elemento. Devuelve **None** si el elemento no se encuentra.

insert(L,element,position)

Descripción: Inserta un elemento en una posición determinada de la lista que representa el **TAD secuencia.**
Entrada: la lista sobre el cual se quiere realizar la inserción (**Linkedlist**) y el valor del elemento (**element**) a insertar y la posición (**position**) donde se quiere insertar.
Salida: Si pudo insertar con éxito devuelve la posición donde se inserta el elemento. En caso contrario devuelve **None**. Devuelve **None** si la posición a insertar es mayor que el número de elementos en la lista.

delete(L,element)

Descripción: Elimina un elemento de la lista que representa el **TAD secuencia.**
Poscondición: Se debe desvincular el **Node** a eliminar.
Entrada: la lista sobre el cual se quiere realizar la eliminación (Linkedlist) y el valor del elemento (element) a eliminar.

Salida: Devuelve la posición donde se encuentra el elemento a eliminar. Devuelve **None** si el elemento a eliminar no se encuentra.

length(L)

Descripción: Calcula el número de elementos de la lista que representa el **TAD secuencia.**
Entrada: La lista sobre la cual se quiere calcular el número de elementos.
Salida: Devuelve el número de elementos.

access(L,position)

Descripción: Permite acceder a un elemento de la lista en una posición determinada.
Entrada: La lista (LinkedList) y la **position** del elemento al cual se quiere acceder.
Salida: Devuelve el valor de un elemento en una **position** de la lista, devuelve **None** si no existe elemento para dicha posición.

update(L,element,position)

Descripción: Permite cambiar el valor de un elemento de la lista en una posición determinada
Entrada: La lista (**LinkedList**) y la **position** sobre la cual se quiere asignar el valor de **element.**
Salida: Devuelve **None** si no existe elemento para dicha posición. Caso contrario devuelve la posición donde pudo hacer el update.

A tener en cuenta:

1. Cada operación básica debe ser implementada como una función.
   Ejemplo:

```
def insert(LinkedList,element,position):
      Aca va el código que implementa la operación insert
```

2. Las operaciones deben respetar la especificación propuesta. Es decir sólo incluir los parámetros mencionados en la definición de la operación.
3. Se sugiere usar lápiz y papel primero.
4. **No se puede utilizar otra Biblioteca mas allá de algo1.py y el módulo previamente desarrollado myarray.py**

Ejercicio 2:

Calcular el orden de complejidad $O(f)$ para cada una de las operaciones básicas del ejercicio 1 y compararlas con las operaciones que se correspondan realizadas sobre arreglos.