De Faveri Tomas

TAD-LISTAS

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

class LinkedList:
    head = None


def mostrarLinkedList(L):
    current = L.head
    print("LINKED LIST: [", end="")
    while current != None:
        if current.nextNode != None:
            print(str(current.value) + ", ", end="")
        else:
            print(str(current.value), end="")
        current = current.nextNode
    print("]")


"""
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

[1, 321, 34]

[12, 1, 321, 34]

"""

def add(L,element):
    newNode = Node()
    newNode.value = element

    newNode.nextNode = L.head
```

```
    L.head = newNode

    #mostrarLinkedList(newLinkedList)

    return 0 #el 0 hace referencia a la posicion


"""
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.
"""

def search(L, element):
    current = L.head
    position = 0
    while current != None:
        if current.value == element:
            return position
        current = current.nextNode
        position += 1

    return None

"""
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.
"""

def length(L):
    current = L.head
    length = 0
    while current != None:
```

```
        length += 1
        current = current.nextNode

    return length

"""
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.
"""

def insert(L,element,position):
    if position <= length(L) and position >= 0:
        if position == 0:
            add(L,element)
        else:
            currentNode = L.head
            for i in range(0, position - 1):
                currentNode = currentNode.nextNode

            newNode = Node()
            newNode.value = element
            newNode.nextNode = currentNode.nextNode
            currentNode.nextNode = newNode
        return position

    return None


"""
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.
"""

def delete(L,element):
    position = search(L,element)
    if position != None:

        if position == 0:
            L.head = L.head.nextNode
        else:
            currentNode = L.head
            for i in range(0, position - 1):
                currentNode = currentNode.nextNode
            currentNode.nextNode = currentNode.nextNode.nextNode

        return position
    else:
        return None


"""
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.
"""

def access(L,position):
    #print(position)
    if (position <= length(L) and position >= 0):

        if (position == 0):
            return L.head.value
        else:
            currentNode = L.head.nextNode
            for i in range(0, position-1):
```

```
                currentNode = currentNode.nextNode

            return currentNode.value

    else:
        return None


"""
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.
"""

def update(L,element,position):
    if position < length(L) and position >= 0:
        if position == 0:
            L.head.value = element
        else:
            currentNode = L.head.nextNode
            for i in range(0, position - 1):
                currentNode = currentNode.nextNode
            currentNode.value = element

        return position
    else:
        return None

    return
```

```
----------------------------------------------------------------------
Ran 17 tests in 0.000s

OK
PS C:\Users\Usuario\Desktop\Algo 1 python>
```