S1-Listas Enlazadas

Insertar al inicio de una lista enlazada - Básico

Insertar al inicio de una lista enlazada - Básico

Ficheros requeridos: insertar.py (Descargar)
Número máximo de ficheros: 3
Tipo de trabajo: Individual

Dada las cabeza de una lista enlazada, agregar un nuevo elemento al inicio y, después, retornar la nueva cabeza de la lista. En caso tal de que la lista esté vacía, retornar una nueva lista con un solo elemento, el nuevo elemento al inicio.

Entrada

No hay que recibir nada por pantalla, es decir, por input.
La función recibe la cabeza de la lista como parámetro.

Salida

Retornar la cabeza de la lista modificada

Ejemplo 1 (Representación gráfica de cómo se ve):

Input:
1 -> 2 -> 3 -> 4 -> 5-> None y 6
Output:
6 -> 1 -> 2 -> 3 -> 4 -> 5-> None

Ejemplo 2 (Representación gráfica de cómo se ve):

Input:
1 -> 2 -> 99 -> 4 -> 5 -> None y 1
Output:
1 -> 1 -> 2 -> 99 -> 4 -> 5 -> None

Ejemplo 3 (Representación gráfica de cómo se ve):

Input:
None y 3
Output:
3 -> None

Ficheros requeridos

insertar.py

```python
class Node:
    def __init__(self, val : int):
        self.val = val
        self.next = None

def insertarAlInicio(head: Node, valor: int) -> Node:
    # El codigo va aqui!
```

Buscar en una lista enlazada con recursión - Básico

Buscar en una lista enlazada con recursión - Básico

Ficheros requeridos: buscar.py (Descargar)
Número máximo de ficheros: 3
Tipo de trabajo: Individual

Dada la cabeza de una lista enlazada, hallar la posición del valor **k** y retornarlo, usando recursión. En caso tal de que la lista esté vacía o no se encuentre el valor en la lista retornar -1.

Entrada

No hay que recibir nada por pantalla, la función dada tiene la cabeza de la lista a recibir como parámetro y el valor **k** a buscar

Salida

Imprimir la primera posición del valor ***k*** del nodo. Este está indexado en 0. Si el elemento no está en la lista enlazada debe imprimir -1.

Es decir, cuenta de la siguiente forma: 0,1,2,3...

Ejemplo 1 (Representación gráfica de cómo se ve):

Input:
1 -> 2 -> 3 -> 4 -> 5->None 4
Output:
3

Ejemplo 2 (Representación gráfica de cómo se ve:

Input:
1 -> 2 -> 99 -> 4 -> 5->None 5
Output:
4

Ejemplo 3 (Representación gráfica de cómo se ve):

Input:
5->3->4->None 9
Output:
-1

Ficheros requeridos

buscar.py

```
class Node:
    def __init__(self, val):
        self.val = val
        self.next = None

def buscar(head: Node, k : int) -> int:
    # El codigo va aqui!
```

Máximo de una lista enlazada con recursión - Básico

Máximo de una lista enlazada con recursión - Básico

Ficheros requeridos: maximo.py (Descargar)
Número máximo de ficheros: 3
Tipo de trabajo: Individual

Dada la cabeza de una lista enlazada, hallar el valor máximo y retornarlo, usando recursión. En caso tal de que la lista esté vacía, retornar 0.

Entrada

No hay que recibir nada usando input.

La función recibe la cabeza de la lista a recibir como parámetro.

Salida

No hay que imprimir nada usando print.

Retornar el valor máximo de la lista

Ejemplo 1 (Representación gráfica de cómo se ve):

Input:
1 -> 2 -> 3 -> 4 -> 5 -> None
Output:
5

Ejemplo 2 (Representación gráfica de cómo se ve):

Input:
1 -> 2 -> 99 -> 4 -> 5 -> None
Output:
99

Ejemplo 3 (Representación gráfica de cómo se ve):

Input:
None
Output:
0

Ficheros requeridos

maximo.py

```
class Node:
    def __init__(self, val : int):
        self.val = val
        self.next = None

def maximo(head: Node) -> int:
    # El codigo va aqui!
```

S1 - Obtener un elemento en la posición i de una lista enlazada - Sencillo

S1 - Obtener un elemento en la posición i de una lista enlazada - Sencillo

Ficheros requeridos: posicion.py (Descargar)
Número máximo de ficheros: 3
Tipo de trabajo: Individual

Dada la cabeza de una lista enlazada, hallar el valor en la posición ***i*** de una lista enlazada y retornarlo. En caso tal de que la lista esté vacía, retornar 0.

Entrada

No hay que recibir nada, la función dada tiene la cabeza de la lista a recibir como parámetro.

Salida

Retornar la cabeza de la lista modificada indexado en 1. Sólo como un ejercicio porque en la vida real se indexan desde 0.

Ejemplo 1 (Representación gráfica de cómo se ve):

Input:
1 -> 2 -> 3 -> 4 -> 5 -> None 1
Output:
1

Ejemplo 2 (Representación gráfica de cómo se ve):

Input:
1 -> 10-> 99 -> 4 -> 11-> None 5
Output:
11

Ejemplo 3 (Representación gráfica de cómo se ve):

Input:
None
Output:
0

Ficheros requeridos

posicion.py

```python
class Node:
    def __init__(self, val):
        self.val = val
        self.next = None

def posicion(head: Node, i: int) -> int:
    # El codigo va aqui!
```

S1 - Insertar un elemento en la posición i de una lista enlazada - Sencillo

S1 - Insertar un elemento en la posición i de una lista enlazada - Sencillo

Ficheros requeridos: insertar.py (Descargar)
Número máximo de ficheros: 3
Tipo de trabajo: Individual

Dada las cabeza de una lista enlazada, insertar un elemento (Nodo) en la posición ***i*** de una lista enlazada. Comienza a contar las posiciones en 1.

Entrada

No hay que recibir nada, la función dada tiene la cabeza de la lista a recibir como parámetro, el valor del Nodo ***N*** *y la posición* ***i*** *en la cual debe ser ingresada* .

Salida

Retornar la cabeza de la lista modificada.

Si la lista está vacía retornar la cabeza sin modificar.

Nota: Si la posición i dada es mayor a la longitud de la lista, entonces retornar la cabeza de la lista dada. Es decir, se pueden agregar elementos hasta la última posición de la lista enlazada. (ver ejemplo 3)

Ejemplo 1 (Representación gráfica de cómo se ve):

Input:
1 -> 2 -> 3 -> 4 -> 5->None 10 1
Output:
10 ->1 -> 2 -> 3 -> 4 -> 5->None

Ejemplo 2 (Representación gráfica de cómo se ve):

Input:
1 -> 2 -> 3 -> 4 -> 5->None 10 8
Output:
1 -> 2 ->3 -> 4 -> 5->None

Ejemplo 3 (Representación gráfica de cómo se ve):

Input:
1 -> 2 -> 3 -> 4 -> 5->None 20 6
Output:
1 -> 2 -> 3 -> 4 -> 5 -> 20->None

Ficheros requeridos

insertar.py

```
class Node:
    def __init__(self, val):
        self.val = val
        self.next = None

def insertar(head: Node, valor, posI):
    # El codigo va aqui!
```

S1 - Borrar el elemento en la posición i de una lista enlazada- Sencillo

S1 - Borrar el elemento en la posición i de una lista enlazada-Sencillo

Ficheros requeridos: borrar.py (Descargar)
Número máximo de ficheros: 3
Tipo de trabajo: Individual

Dada la cabeza de una lista enlazada, borrar el elemento en la posición ***i*** y retornar la cabeza. En caso tal de que la lista esté vacía, retornar la cabeza.

Entrada

No hay que recibir nada, la función dada tiene la cabeza de la lista a recibir como parámetro.

Salida

Retornar la cabeza de la lista modificada. 0≤ i ≤ n. Donde ***i*** es la posición del nodo a eliminar y ***n*** es la longitud de la lista enlazada.

Nota: La lista está indexada en 1. Es decir, empieza a contar desde 1

Ejemplo 1 (Representación gráfica de cómo se ve):

Input:
1 -> 2 -> 3 -> 4 -> 5. 1
Output:
2-> 3 -> 4 -> 5

Ejemplo 2 (Representación gráfica de cómo se ve):

Input:
1 -> 2 -> 3 -> 4 -> 5. 5
Output:
1-> 2 -> 3 -> 4

Ejemplo 3 (Representacion gráfica de cómo se ve):

Input:
Output:
0

Ficheros requeridos

borrar.py

```
class Node:
    def __init__(self, val):
        self.val = val
        self.next = None

def borrar(head: Node, pos):
```

S1 - Insertar un elemento en la posición i de una lista enlazada con recursión- Intermedio

S1 - Insertar un elemento en la posición i de una lista enlazada con recursión- Intermedio

Ficheros requeridos: insertar.py (Descargar)
Número máximo de ficheros: 3
Tipo de trabajo: Individual

Dada las cabeza de una lista enlazada, insertar un elemento (Nodo) en la posición ***i*** de una lista enlazada de forma RECURSIVA.

Entrada

No hay que recibir nada, la función dada tiene la cabeza de la lista a recibir como parámetro, el valor del Nodo ***N*** *y la posición* ***i*** *en la cual debe ser ingresada* .

Salida

Retornar la cabeza de la lista modificada.

Si la lista está vacía retornar la cabeza sin modificar.

Nota: Si la posición i dada es mayor a la longitud de la lista, entonces retornar la cabeza de la lista dada. Es decir, se pueden agregar elementos hasta la última posición de la lista enlazada. (ver ejemplo 3)

Ejemplo 1 (Representación gráfica de cómo se ve):

Input:
1 -> 2 -> 3 -> 4 -> 5->None 10 1
Output:
10 ->1 -> 2 -> 3 -> 4 -> 5->None

Ejemplo 2 (Representación gráfica de cómo se ve):

Input:
1 -> 2 -> 3 -> 4 -> 5->None 10 8
Output:
1 -> 2 ->3 -> 4 -> 5->None

Ejemplo 3 (Representación gráfica de cómo se ve):

Input:
1 -> 2 -> 3 -> 4 -> 5->None 20 6
Output:
1 -> 2 -> 3 -> 4 -> 5 -> 20->None

Ficheros requeridos

insertar.py

```python
class Node:
    def __init__(self, val):
        self.val = val
        self.next = None

def insertar(head: Node, valor, posI):
    # El codigo va aqui!
```

S1 - Invertir una lista simplemente enlazada - Intermedio

S1 - Invertir una lista simplemente enlazada - Intermedio

Ficheros requeridos: invertir.py (Descargar)
Tipo de trabajo: Individual

Dada la cabeza de una lista simplemente enlazada, devolver la lista invertida (reversada).

No, no hay crear listas de ejemplo. Moodle se encargará de las pruebas. Sólo hay que definir la función.

Entrada

No hay que recibir nada por pantalla. La función recibe la cabeza de la lista a recibir como parámetro.

Salida

No hay que imprimir nada por pantalla. La función debe retornar la nueva cabeza de lista.

Ejemplo 1 (Representación gráfica de cómo se ve):

Input:
1 -> 2 -> 3
Output:
3 -> 2 -> 1

Ficheros requeridos

invertir.py

```
class Node:
    def __init__(self, val):
        self.val = val
        self.next = None

def invertir(head: Node) -> Node:
    # El codigo va aqui!
```

S1 - Rotar la lista - Intermedio

S1 - Rotar la lista - Intermedio

Ficheros requeridos: rotar.py (Descargar)
Tipo de trabajo: Individual

Dada la cabeza de una lista enlazada simple y un numero N, devuelve la cabeza de la lista modificada despues de rotar la lista hacia la derecha N veces.

Cuando un nodo esta al final, este llega al comienzo.

Ej:

1->2->3

Pasa a ser

3->1->2

Entrada

No hay que recibir nada, la funcion dada tiene la cabeza de la lista a recibir como parametro y el valor N, el numero de veces a rotar la lista.

Salida

Retornar la cabeza de la lista modificada/rotada

Ejemplo 1 (Representacion grafica de como se ve):

Input:
1 -> 2->3 -> 4 -> 5 N=3
Output:
3 -> 4 -> 5->1 -> 2

S1 - Finanzas - Intermedio

S1 - Finanzas - Intermedio

Ficheros requeridos: finanzas.py (Descargar)
Tipo de trabajo: Individual

Mauricio, un estudiante de estructuras de datos 1 ha decidido crear un programa que le ayude con sus propias finanzas, para ello decidió que el monto de sus ingresos va a estar representado en listas enlazadas, de modo que cada dígito del monto es un elemento de la lista, sin embargo, Mauricio tiene un gran problema, y es que al tener estos valores en listas no sabe cómo calcular el total de dinero que ha ganado en el mes. Para ayudar a Mauricio debes crear un programa que pueda hacer la suma de los distintos ingresos que él tuvo en el mes (a Mauricio le pagan cada 15 días por lo que solo tiene dos veces ingresos en el mes) y NO mostrarlo en la pantalla, NI retornarlo como un entero, sino retornarlo en tipo cadena de caracteres.

Ejemplo:

Entrada:

Lista1= 2→9→0→0→0→0→0

Lista2= 3→1→0→0→0→0→0

Salida:

'6000000'

Ficheros requeridos

finanzas.py

```
class Node:
    def __init__(self, val):
        self.val = val
        self.next = None

def sumar(head1: Node, head2: Node):
    #codigo aqui
```

S1 - Biblioteca - Intermedio

S1 - Biblioteca - Intermedio

Ficheros requeridos: ordenar.py (Descargar)
Tipo de trabajo: Individual

En una biblioteca hay n libros representados en una lista enlazada, pero por error fueron almacenados de forma incorrecta y ahora la bibliotecaria no es capaz de encontrar rápidamente el que necesita, ayúdala a resolver este problema ordenando los libros por su año de lanzamiento y si dos años son iguales ordenarlos por su nombre. Completa la función ordenar y retorna la cabeza de la lista.

Ejemplo

Entrada

{1980,"Encanto"} -> {1975,"Sueños de un paraíso"} -> {2000,"Nunca me olvides"}

Salida

{1975,"Sueños de un paraíso"} -> {1980,"Encanto"} -> {2000,"Nunca me olvides"}

Ficheros requeridos

ordenar.py

```
class Node:
    def __init__(self, fecha,nombre):
        self.fecha = fecha
        self.nombre = nombre
        self.next = None

def ordenar(head:Node):
    #codigo aqui
```

S1 - Teclado Roto - Intermedio

S1 - Teclado Roto - Intermedio

Ficheros requeridos: kb.py (Descargar)
Tipo de trabajo: Individual

Enunciado
Una vez redactando un texto para la universidad te das cuenta de que tu teclado no está funcionando, está dañado!
Después de jurar no volver a comprar nada por Wish, te das cuenta de que realmente no está tan malo, solo que a veces se presionan automáticamente las teclas de Home y End, los cuales te envían al inicio o al fin de un texto respectivamente.

Como estabas escribiendo el ensayo a las 2 A.M con 3 cafés y una Monster encima, no te diste cuenta de que esto estaba sucediendo, pero ahora que estas más lucido revisas lo que escribiste y tu tarea es encontrar como quedo el texto después de estos movimientos.

Problema
Dado un string, compuesto por números, barras bajas, letras y dos caracteres especiales "[" y "]", (Donde "[" representa presionar Home y "]" representa presionar End), encontrar como se ve el texto después de ser afectado por los movimientos al inicio y fin del texto.

Entrada

Un string antes de ser afectado por los movimientos de Home y End

Salida

La cabeza de una lista simplemente enlazada donde en cada nodo va un solo caracter. Es decir, no van varios caracteres en un nodo.

Ejemplo 1 (Representación gráfica de como se ve):

Input:
This_is_a_[Beiju]_text [[]][[]]Happy_Birthday_to_Tsinghua_University
Output:
BeijuThis_is_a__text Happy_Birthday_to_Tsinghua_University

Explicación:
This_is_a_[Beiju]_text
En este ejemplo, se alcanza a copiar
- This_is_a_
Luego se va al inicio del texto al presionar [y se copia Beiju
- BeijuThis_is_a_
Luego se presiona], vamos al final del texto y pegamos _text
- BeijuThis_is_a__text

Ficheros requeridos

kb.py

```python
class Node:
    def __init__(self, val):
        self.val = val
        self.next = None

def broken(s: str):
    # El codigo va aqui!
    # Se retorna la cabeza de la lista
    # (Un objeto nodo)
```

S1 - Aplana una lista (con recursión) - Demandante

S1 - Aplana una lista (con recursión) - Demandante

Ficheros requeridos: aplanar.py (Descargar)
Tipo de trabajo: Individual

En la vida real, al modelar sistemas de información, es común que tengamos una lista de personas, donde cada persona --a su vez-- es un nombre y una lista de compras... y cada compra es una lista de productos y así, tantos niveles como sea necesario.

Dada la cabeza de una lista simplemente enlazada recibida por parámetro, debes aplanar la lista y retornarla. NO imprimirla.
Una lista aplanada es una lista donde cada elemento es un número y NO es otra lista.

Ten en cuenta, que la lista enlazada simple que recibes puede ser una lista de números, una lista de listas de números, una lista de listas de listas de números o una lista de lista de lista de lista... de números, sin una limitación en los niveles.
Para poder representar este tipo de listas enlazadas, en el atributo valor (*val*), en algunos casos, habrá un entero y en otros casos será un nodo que representa que un nodo tiene otro nodo (es decir, una sublista) como atributo.

NO es posible agregar otros atributos a la clase *Node*

NO se puede modificar la lista original que reciben como parámetro

NOTA: En Python, para saber si una *variable* es de tipo *Node* usamos el comando *isinstance(variable, Node)*.

Entrada

NO hay que recibir nada con *input()*, la función recibe la cabeza de la lista como parámetro.
La entrada NO es una lista de python (en inglés, *list*).

Salida

NO hay que imprimir la lista, la función debe retornar la cabeza de la lista aplanada
La salida NO es una lista de python (en inglés, *list*).

Ejemplo 1 -- Representación gráfica de cómo se ve usando imprimirListaEnlazada(cabeza) :

Input:
(1->None)->((3->None)->(4->None)->None)->5->None
Output:
1->3->4->5->None

Ejemplo 1 -- Representación gráfica de cómo se vería usando la notación de corchetes:

Input:
[[1], [[3], [4]], 5]
Output:
[1,3,4,5]

Ejemplo 1 -- Representación con código en Python:

Input:

```
Node1 = Node(1)

Node3 = Node(3)

Node4 = Node(4)

Node5 = Node(5)

NodeLista1 = Node(Node1)

NodeLista3 = Node(Node3)

NodeLista4 = Node(Node4)

NodeLista3.next = NodeLista4

NodeLista3y4 = Node(NodeLista3)

NodeLista1.next = NodeLista3y4

NodeLista3y4.next = Node5

cabeza = NodeLista1
```

Output:

```
Node1 = Node(1)
Node3 = Node(3)
Node4 = Node(4)
Node5 = Node(5)

Node1.next = Node3
Node3.next = Node4
Node4.next = Node5


cabeza = Node1
```

Ficheros requeridos

aplanar.py

```
class Node:
    def __init__(self, val):
        self.val = val
        self.next = None

def imprimirListaEnlazada(head: Node):
    """Este método se usa para depuración solamente"""
    if head == None:
       print("None",end='')
    else:
        if isinstance(head.val,Node):
          print("(",end='')
          imprimirListaEnlazada(head.val)
          print(")",end='->')
        else:
          print(head.val,end='->')
        imprimirListaEnlazada(head.next)


def aplanar(head: Node):
    #¡El código va aquí!
```