Historial de navegación

Un navegador web mantiene dos pilas internas para gestionar el historial de navegación: una “hacia atrás” (back) y otra “hacia adelante” (forward).

Cuando el usuario navega a una nueva página, ésta se añade al final de la deque; al pulsar “atrás”, la página actual pasa al frente de la deque de forward, y la anterior queda como actual.

Una deque (cola de doble extremo) permite inserciones y eliminaciones en ambos extremos en tiempo O(1).

El dueño del producto “Navegador” junto con el equipo de desarrollo logran definir los requerimientos del programa utilizando la siguiente historia de usuario.

Historia de usuario Nro.	1	**Título:**	Gestionar historial de navegación
Descripción	COMO:	Usuario de un navegador web	
	QUIERO:	Navegar hacia atrás y adelante en mi historial de páginas visitadas	
	PARA:	Recuperar páginas recientes sin recargar manualmente	
Criterios de aceptación	▪ Se debe poder visitar una nueva página (VISIT <URL>) que vacíe la cola de “forward” y añada la URL al final de la deque de “back”. ▪ Al ejecutar BACK, la URL actual se mueve al frente de “forward” y la última de “back” pasa a ser actual. ▪ Al ejecutar FORWARD, la URL actual se mueve al final de “back” y la primera de “forward” pasa a ser actual. ▪ Si no hay dónde retroceder o avanzar, se imprime "IGNORAR".		

Usted es contratado por el equipo de desarrollo para construir un programa en python que cumpla las funcionalidades requeridas por el product owner teniendo como referencia las historias de usuario presentadas previamente.

Entrada	Varias líneas con comandos: ▪ VISIT <url> ▪ BACK ▪ FORWARD ▪ END (termina la entrada)
Salida	Por cada BACK, FORWARD o VISIT, imprimir la URL actual tras la operación, o "IGNORAR" si la operación no cambia la página.

Instrucciones para la calificación automática

Antes de enviar la solución del reto, por favor tenga en cuenta los siguientes aspectos:

- La clase principal debe llamarse RetoSecuencial.
- Debe tener un método run.
- El programa lee de la entrada estándar hasta END.
- En cada comando imprime únicamente la URL actual o "IGNORAR".
- Únicamente debe imprimir los resultados finales. No imprima valores con cálculos intermedios

Casos de prueba (visibles)

Entradas de ejemplo	VISIT a.com VISIT b.com BACK BACK FORWARD END
Salida de ejemplo	a.com b.com a.com IGNORAR b.com

Casos de prueba (ocultos)

Caso prueba	Entrada	Salida esperada
1	BACK VISIT c.com FORWARD BACK END	IGNORAR c.com IGNORAR IGNORAR

Código Python :

```
from collections import deque



class RetoSecuencial:

    def run(self):
```

```
back = deque()
forward = deque()
current = None

while True:
    try:
        line = input().strip()
        if line == "END":
            break
        parts = line.split(maxsplit=1)
        cmd = parts[0]

        if cmd == "VISIT" and len(parts) == 2:
            if current:
                back.append(current)
            current = parts[1]
            forward.clear()
            print(current)

        elif cmd == "BACK":
            if back:
                forward.appendleft(current)
                current = back.pop()
                print(current)
            else:
```

```
                    print("IGNORAR")


            elif cmd == "FORWARD":
                if forward:
                    back.append(current)
                    current = forward.popleft()
                    print(current)
                else:
                    print("IGNORAR")
            except Exception:
                continue


if __name__ == "__main__":
    RetoSecuencial().run()
```

```python
if __name__ == "__main__":
    RetoSecuencial().run()
```

```
VISIT a.com
a.com
VISIT b.com
b.com
BACK
a.com
BACK
IGNORAR
FORWARD
b.com
END
```

```python
if __name__ == "__main__":
    RetoSecuencial().run()
```

```
BACK
IGNORAR
VISIT c.com
c.com
FORWARD
IGNORAR
BACK
IGNORAR
END
```