Editor de texto

Sofía está desarrollando un editor de texto con deshacer automático. Cada vez que se realiza una acción (escribir, borrar, pegar), se almacena en una pila para poder deshacerla en orden inverso.

Una estructura LIFO (Last In, First Out) como la pila es perfecta para este caso.

Sofía logra definir los requerimientos del programa utilizando las siguientes historias de usuario.

Historia de usuario Nro.	1	**Título:**	Deshacer acciones en un editor de texto
Descripción	COMO:	Desarrolladora de un editor de texto	
	QUIERO:	Poder deshacer las últimas acciones realizadas	
	PARA:	Brindar al usuario la opción de revertir errores o cambios indeseados	
Criterios de aceptación	▪ Debe poder agregarse una acción a la pila (PUSH <acción>). ▪ Debe poder deshacerse la última acción (POP). ▪ Si se intenta deshacer y la pila está vacía, debe imprimirse "PILA VACIA". ▪ Las acciones se imprimen al ser deshechas (POP).		

Usted es contratado por Sofia para construir un programa en Python que cumpla las funcionalidades requeridas por Sofia teniendo como referencia las historias de usuario presentadas previamente.

Entrada	Varias líneas con comandos: ▪ PUSH <acción> ▪ POP ▪ END (finaliza la entrada)
Salida	Por cada POP, se imprime la acción eliminada, o "PILA VACIA" si la pila está vacía.

Instrucciones para la calificación automática

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

- Cada caso de prueba termina con END.
- El código debe procesar los comandos uno por uno.
- Debe existir una clase llamada RetoPila.
- En ella debe existir un método run.
- El código debe leer desde la entrada estándar y escribir en la salida estándar.
- Solo debe imprimirse la salida correspondiente a POP.

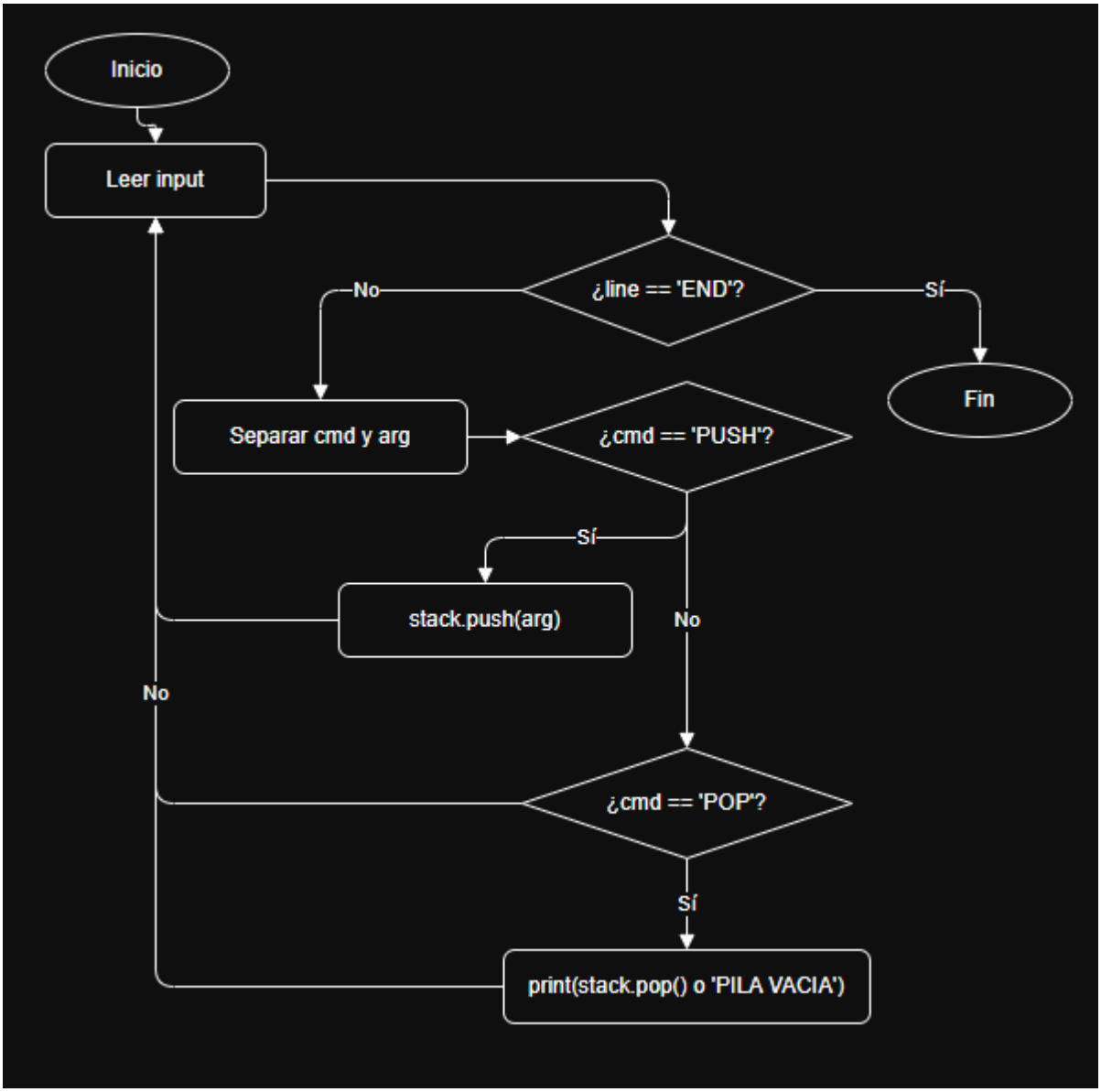
Casos de prueba (visibles)

Entradas de ejemplo	PUSH Escribir Hola PUSH Borrar Hola POP POP POP END
Salida de ejemplo	Borrar Hola Escribir Hola PILA VACIA

Casos de prueba (ocultos)

Caso prueba	Entrada	Salida esperada
1	POP PUSH Pegar imagen POP END	PILA VACIA Pegar imagen
2	PUSH Cortar párrafo PUSH Escribir título PUSH Pegar párrafo POP POP END	Pegar párrafo Escribir título

Diagrama de flujo: Diseño



Código Python :

```
class Stack:

    def __init__(self):

        self.items = []



    def push(self, item):

        self.items.append(item)
```

```
    def pop(self):
        if not self.items:
            return None
        return self.items.pop()


class RetoPila:
    def run(self):
        stack = Stack()
        while True:
            try:
                line = input()
                if line.strip() == "END":
                    break
                parts = line.strip().split(maxsplit=1)
                command = parts[0]


                if command == "PUSH" and len(parts) == 2:
                    stack.push(parts[1])
                elif command == "POP":
                    result = stack.pop()
                    print(result if result else "PILA VACIA")
            except Exception:
                continue


if __name__ == "__main__":
```

```
RetoPila().run()
```

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

```
PUSH Escribir Hola
PUSH Borrar Hola
POP
Borrar Hola
POP
Escribir Hola
POP
PILA VACIA
END
```

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

```
POP
PILA VACIA
PUSH Pegar imagen
POP
Pegar imagen
END
```

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

```
PUSH Cortar párrafo
PUSH Escribir título
PUSH Pegar párrafo
POP
Pegar párrafo
POP
Escribir título
END
```