A. Calcula la complejidad de las siguientes secuencias de código:

1.

```
def isEven(value):
    if value % 2 == 0
        return True
    else:
        return False

print(isEven(n))
```

2.

```
a = 0
n = len(list1)
for x in list1:
    a += min(list1)
print(a)
```

3.

```
def recur(n):
    if n in (1,2,3):
        return 1
    return sum(recur(n-i) for i in range(1,4))

print(recur(N))
```

4.

```
c = 1
for i in range(n):
    for j in range(i,2*n):
        c *= 4
print(c)
```

5.

```
def permutation(array, start, result)
    if (start == len(array)):
        result.append(array[:])

    for i in range(start,len(array)-1):
        array[start], array[i] = array[i], array[start]
        permutation(array, start+1,result)
        array[start], array[i] = array[i], array[start]

    return result

print(permutation(list1, 0, [])
```

B. Dada una cadena de longitud ≤ 8 que formada por caracteres ‘I’ y ‘D’, donde ‘I’ denota la secuencia creciente y ‘D’ denota la secuencia decreciente, descifra la secuencia para construir el número mínimo sin cifras repetidas usando la estructura de la pila.

cadena		**salida**
IIDDIDID	⟶	125437698
IDIDII	⟶	1325467
DDDD	⟶	54321
IIII	⟶	12345

C. ¿Cuál es la salida para las siguientes secuencias de código?

1.

```
q = Cola()
q.enqueue(1)
q.enqueue(1)

for _ in range(10):
    a = q.dequeue()
    print(a)
    b = q.peek()
    q.enqueue(a+b)
```

2.

```
q = Cola()
s = Pila()

for i in range(3):
    q.enqueue(i+1)

for i in range(3):
    s.push(-i-1)

for _ in range(7):
    a = q.dequeue()
    b = s.pop()

    print(a-b)
    q.enqueue(a+b)
    s.push(a+b)
```

D. Se te dan dos arrays (a1, a2, ..., an) y (b1, b2, ..., bn).

En una operación, puedes elegir cualquier entero i tal que 1 ≤ i ≤ n e intercambiar los números ai y bi.

Determina si, después de usar cualquier número de operaciones (posiblemente cero), se pueden satisfacer simultáneamente las siguientes dos condiciones:

an = max(a1, a2, ..., an),
bn = max(b1, b2, ..., bn).

Aquí, max(c1, c2, ..., ck) denota el número máximo entre c1, c2, ..., ck. Por ejemplo, max(3, 5, 4) = 5, max(1, 7, 7) = 7, max(6, 2) = 6.

Entrada:
Cada prueba contiene múltiples casos de prueba. La primera línea contiene el número de casos de prueba t (1 ≤ t ≤ 200). A continuación, sigue la descripción de los casos de prueba.

La primera línea de cada caso de prueba contiene un solo entero n (1 ≤ n ≤ 100) — la longitud de los arrays.

La segunda línea de cada caso de prueba contiene n enteros a1, a2, ..., an (1 ≤ ai ≤ 100) — elementos del primer array.

La tercera línea de cada caso de prueba contiene n enteros b1, b2, ..., bn (1 ≤ bi ≤ 100) — elementos del segundo array.

Salida:
Para cada caso de prueba, imprime "Yes" si, después de usar cualquier número de operaciones, se satisfacen las condiciones descritas anteriormente. De lo contrario, imprime "No".

Ejemplo:
Entrada:
3
3
7 9 7
7 6 9
4
10 10 15 15
10 16 15 15
2
100 99
99 100

Salida:
Yes
No
Yes

E. Una secuencia de n números se llama permutación si contiene todos los enteros del 1 al n exactamente una vez. Por ejemplo, las secuencias [3,1,4,2], [1] y [2,1] son permutaciones, pero [1,2,1], [0,1] y [1,3,4] no lo son.

Fernando tenía una permutación p de n elementos. La escribió en la pizarra n veces de la siguiente manera:

Mientras escribía la permutación en la i-ésima ($1 \le i \le n$) vez, omitía el i-ésimo elemento. Por lo tanto, escribió un total de n secuencias de longitud $n-1$ cada una.

Por ejemplo, supongamos que Fernando tenía una permutación p = [4,2,1,3] de longitud 4. Entonces hizo lo siguiente:

Escribió la secuencia [2,1,3], omitiendo el elemento $p_1=4$ de la permutación original.
Escribió la secuencia [4,1,3], omitiendo el elemento $p_2=2$ de la permutación original.
Escribió la secuencia [4,2,3], omitiendo el elemento $p_3=1$ de la permutación original.
Escribió la secuencia [4,2,1], omitiendo el elemento $p_4=3$ de la permutación original.

Conoces todas las n secuencias que se han escrito en la pizarra, pero no conoces el orden en el que fueron escritas. Se dan en un orden arbitrario. Reconstruye la permutación original a partir de ellas.

Por ejemplo, si conoces las secuencias [4,2,1], [4,2,3], [2,1,3], [4,1,3], entonces la permutación original será p = [4,2,1,3].

Entrada:
La primera línea de los datos de entrada contiene un solo número entero t ($1 \le t \le 10000$) — el número de casos de prueba.

A continuación, sigue la descripción de los casos de prueba.

La primera línea de cada caso de prueba contiene un solo número entero n ($3 \le n \le 100$).

Después de esto, hay n líneas, cada una conteniendo exactamente $n-1$ enteros y describiendo una de las secuencias escritas en la pizarra.

Salida:
Para cada caso de prueba, imprime en una línea separada una permutación p de manera que las n secuencias dadas puedan obtenerse a partir de ella.

Ejemplo:
Entrada:
3
4
4 2 1
4 2 3
2 1 3
4 1 3
3
2 3
1 3
1 2
5
4 2 1 3
2 1 3 5
4 2 3 5
4 1 3 5
4 2 1 5

Salida:
4 2 1 3
1 2 3
4 2 1 3 5

F. Dado un array de n enteros, tu tarea es encontrar la suma máxima de valores en una subarray contiguo y no vacío.

Entrada
La primera línea de entrada contiene un entero n, que es el tamaño del array.

La segunda línea contiene n enteros x1, x2, ..., xn, que son los valores del array.

Salida
Imprime un entero: la suma máxima del subarray.

Restricciones
$1 \leq n \leq 2 * 10^5$
$-10^9 \leq x_i \leq 10^9$

Ejemplo
Entrada
8
-1 3 -2 5 3 -5 2 2

Salida
9