Context

You are given an **N × N correlation matrix** `C`, where each entry `C[i][j]` represents the correlation between two variables `i` and `j`.
All diagonal entries are 1, and the matrix is symmetric.

You are allowed to **flip the sign of any variable**.
Flipping a variable `i` means multiplying **both** its corresponding **row and column** by −1.

Your goal is to determine which variables should be flipped in order to **maximize the total sum of all correlations** in the resulting matrix.

Example

If `C` were:

$$C = \begin{bmatrix} 1 & -0.8 \\ -0.8 & 1 \end{bmatrix}$$

Then flipping the second variable changes it to:

$$C' = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix}$$

The total sum of entries has increased from `0.4` to `3.6`, so the optimal flip set is `{1}`.

Your Task

Implement an algorithm that:

1. Takes as input a symmetric **correlation matrix** `C` (size N×N).

2. Determines which variables to flip (i.e., which indices to multiply by −1).
3. Returns:
    - The **maximum total correlation sum** achievable.
    - The **set of flipped variable indices** (0-based or 1-based; just be consistent).

Input

You will be provided a NumPy array `C` representing the correlation matrix.

Example (already loaded in your environment):

```
C = np.array([
    [1.00,  0.80, -0.70],
    [0.80,  1.00, -0.65],
    [-0.70, -0.65,  1.00]
])
```

Row = 2, col = 2

Cx = kx

row=i, col=i

C = corr(X_1:n)

Output

Print or return a result in the format:

```
{"max_sum": 7.30,
  "flipped": [2]}
```

Constraints

- Matrix size `N` will be between **6 and 15**
- You may assume `C` is symmetric with 1s on the diagonal.

Evaluation Criteria

- The total correlation of the flipped matrix
- Is the algorithm scalable to large matrix sizes (~1000 x 1000) and beyond