Problem Summary:

Given a list of cellular operators, each characterized by a template that specifies bandwidth (BW), delay, and packet loss, a user submits a request with two requirements: a minimum bandwidth and a desired level of resiliency (High, Medium, or Low).

The goal is to design a resiliency function that evaluates the available operator combinations within the coverage area, ensuring that they collectively satisfy the user's required bandwidth and resiliency level. This function generates all viable combinations of operator networks that can meet these criteria. Once the resiliency function identifies the feasible combinations, an AI Planner assigns the optimal set(s) of operator networks to the user based on the generated options, ensuring their bandwidth and resiliency needs are met.

The following section outlines the rules applied to define the three levels of resiliency. In the next phase of development, we will incorporate latency and packet loss considerations into the logic. Additionally, when allocating bandwidth to a user, we must account for the residual bandwidth available on each operator's network to accommodate subsequent user requests. However, this aspect has not yet been implemented, as we need to determine whether the AI planner and the templates it generates for each request should handle residual bandwidth management, or if this responsibility should fall under the resiliency function. For now, we define the levels of resiliency as follows::

1. **High Resiliency:**
   High resiliency means the network must tolerate failures in all but one of the operators committed to the user. This implies that each operator within every combination returned by the function must independently provide the minimum bandwidth required by the user.

   - The function will return all combinations of operators (with at least two operators in each combination) where each operator in the combination individually meets the user's minimum bandwidth requirement.
   - For example, if there are three operators in the combination, all three must individually satisfy the minimum bandwidth requested by the user.
2. **Medium Resiliency:**
   Medium resiliency ensures that the required bandwidth is still met even if any one operator in the combination fails.

   - The function will return all combinations of operators (with at least two operators in each combination) such that the remaining operators in the combination can meet the bandwidth requirement even if one operator fails.
   - For instance, if a combination includes three operators, any two of the three must collectively meet the required bandwidth. Similarly, if a combination has only two operators, both must work together to satisfy the bandwidth requirement in the event of a failure.
3. **Low Resiliency:**
   Low resiliency assumes that the bandwidth requirement will only be met when all operators in the combination are operational, without any failures.

- The function will return combinations of operators (with at least two operators in each combination) such that the total capacity of all operators in the combination is sufficient to meet the bandwidth requirement.
- For example, if a combination includes two or three operators, the aggregate bandwidth provided by all operators together must satisfy the user's bandwidth requirement.

Subroutine Assumptions:

1. A **list of operators** is provided, where each operator is characterized by three attributes:
   - **Bandwidth** (BW), **Latency** (delay), **Packet Loss**
2. The **user's requirements** include:
   - A **minimum bandwidth** value that must be met.
   - A **resiliency level** (High, Medium, or Low) indicating the tolerance to operator failures.
3. The algorithm will:
   - Analyze the operators and their characteristics.
   - Return combinations of operators that satisfy the user's bandwidth requirement and resiliency level.
4. **Current Scope**:
   - The algorithm focuses only on **bandwidth** and **resiliency** (tolerance to failures).
   - Future iterations of the algorithm may incorporate **latency** (delay) and **packet loss** as additional factors in selecting optimal operator networks.

Subroutine Logic:

Determining Resiliency Combinations

1. **High Resiliency**
   - All operators in a combination must individually meet or exceed the required bandwidth (**Operator**`_BW >= required_BW`).
   - This ensures that the required bandwidth is maintained even if all but one operator in the combination fail.
   - Example: With three operators with BWs [200, 105, 80], If the minimum bandwidth is 100, a valid combination might be `(Operator1(BW=200), Operator2(BW=105))`, as both individually satisfy `BW >= required_BW`.

2. **Medium Resiliency**
   - Include all High Resiliency Combinations (As specified in above case 1) AND
   - At least two operators in a combination must individually meet or exceed the required bandwidth (`BW >= required_BW`), ensuring redundancy.

- At least one operator in the combination must fail to meet the minimum bandwidth (`BW < required_BW`).
- Additionally, combinations where one operator meets `BW >= required_BW` and the remaining operators collectively meet or exceed the minimum bandwidth (`total_BW - single_operator_BW >= required_BW`) are also valid.
- This ensures that the required bandwidth is maintained when one operator fails, as long as others can still satisfy the demand.
- Example: For `required_BW = 100`, valid combinations might include:
  - `(Operator1(BW=110), Operator2(BW=90), Operator3(BW=80))` (one operator meets `BW >= required_BW`, and the others collectively meet `BW >= required_BW`).

3. **Low Resiliency**
   - Include all Medium Resiliency Combinations (As specified in above case 2) AND
   - Includes:
     1. Individual operators that meet or exceed the minimum bandwidth (Operator_`BW >= required_BW`) as standalone valid entries.
     2. Combinations where all operators have bandwidths below the minimum (Operator_`BW < required_BW`), but the **total combined bandwidth** meets or exceeds the minimum (`total_BW >= required_BW`).
   - Combinations with more than one operator having `BW >= required_BW` are excluded.
   - Example: For `required_BW = 60`, valid combinations might include:
     1. `(Operator1(BW=70))` (standalone operator meets `BW >= required_BW`).
     2. `(Operator2(BW=30), Operator3(BW=40))` (total bandwidth `30 + 40 = 70`, and all `BW < required_BW`).

Combination Logic

- The function iterates over all combinations of operators, ranging from size 2 to the total number of operators.
- For each resiliency level, it applies specific conditions to decide whether the combination is valid.
- The logic ensures that:
  - For **High Resiliency**, every operator in the combination is independently capable.
  - For **Medium Resiliency**, redundancy is maintained while allowing at least one operator to fail.
  - For **Low Resiliency**, flexibility is provided by including standalone operators or combinations that collectively meet the requirement without redundancy.