Problem 3: MediSort: Hospital Emergency Queue (Detailed Scenario)

A major metropolitan hospital, **CityCare Emergency Center**, handles a large number of patient walk-ins and ambulance arrivals daily. The hospital uses a triage system to decide **which patient should be treated next**, but due to increasing crowd size and limited doctors, the administration wants to introduce an **algorithmic priority scheduler** called **MediSort**.

MediSort must sort and schedule patients based on three critical factors:

1. **Urgency level** – how critical the patient's condition is (e.g., cardiac arrest > fever).
2. **Waiting time** – how long the patient has already waited in the emergency area.
3. **Severity score** – a numeric score given by triage nurses (higher score = more severe condition).

Each factor contributes differently to a **waiting cost**, and the system must approximate the **minimal total waiting cost** while maintaining fairness and medical ethics.

Scenario Description

During one busy shift, CityCare receives **N patients**:

Each patient P_i has the following attributes:

- **Urgency Category** (U):
  - U1 = Critical
  - U2 = High
  - U3 = Medium
  - U4 = Low
- **Waiting Time (W)**:
  Time (in minutes) since the patient entered triage.
- **Severity Score (S)**:
  A numeric value (0–100), assigned by the triage nurse based on symptoms, vitals, and tests.

To balance these three factors, the hospital assigns **weights**:

- Weight for Urgency = w_U
- Weight for Waiting Time = w_W
- Weight for Severity = w_S

A **priority cost function** is defined as:

$$Cost(P_i) = w_U \cdot U_i + w_W \cdot W_i + w_S \cdot S_i$$

The **higher the cost**, the sooner the patient should be treated.

However, due to unpredictable emergencies, the hospital does not need an exact optimal solution—an **approximation** is acceptable if it closely reflects the minimal waiting cost.

Your Task

Design an algorithm that sorts the entire list of incoming patients to determine their **treatment sequence**, using the following required techniques:

1. Merge Sort (for stable sorting with custom comparator)

Use a modified Merge Sort to:

- compare patients using the combined weighted cost
- ensure stable ordering in case two patients have equal cost
- handle dynamic updates in real-time (merge sort works well with linked structures)

The merge sort comparator must use the cost function above.

2. Weighted Greedy Prioritization

Before full sorting, the system must:

- compute each patient's priority cost
- select a small number of **top candidates** (e.g., 10–20%) using a greedy heuristic
- place them at the front to reduce worst-case waiting

This greedy step acts as a **fast approximation** of minimal waiting cost:

- always choose the "next" patient with maximum cost
- temporarily freeze that position
- then apply merge sort to the remaining list

3. Approximation Strategy

Since emergencies change dynamically:

- exact minimal waiting cost may be infeasible
- instead, approximate by combining:
  - greedy initial shortlist
  - merge-sort-based final ordering
  - continuous recomputation of cost if waiting time increases by threshold amounts

Your algorithm must ensure:

- no patient with critical urgency (U1) appears after any U3 or U4 patients
- no patient waits too long due to equal-cost ties
- total waiting cost ≈ minimal within small approximation error

Expected Output

Your algorithm must produce:

1. Final Ordered Queue of Patients

Sorted list with their priority cost and attributes.

Example:

| Rank | Patient | Urgency | Wait Time | Severity | Priority Cost |

2. Approximation Justification

Explain how close the solution is to minimal cost (qualitative acceptable).

3. Verification

- ordering respects weighted cost
- merge sort is correctly implemented
- greedy front-loading is justified
- stable sorting preserved in equal-priority cases

Marks Distribution

Criteria	**Marks**
Correctness (Sorting + comparison logic)	10
Optimization (Greedy + weighted approximation)	10
Implementation (Merge sort + tie-handling)	10
Total	**30 marks**