## **Midterm Exam Problem Statement**

## Scenario: **Hospital Emergency Queue System**

**By Muhammad Hammad Shakeel From Bsai**

**Midterm Exam – Problem Statement**  
**Scenario: Hospital Emergency Queue System**

A small hospital is managing patients in the Emergency Room (ER). Because patient priority can change quickly (new critical patients arrive, discharged patients leave, etc.), the hospital wants a flexible system to keep track of the current patients in the ER.You are required to design and manipulate this ER Queue using a **Doubly Linked List**.

Each node in the list represents one patient and stores:

* **patientID** (integer)
* Pointer to **previous** patient (prev)
* Pointer to **next** patient (next)

The ER Queue must support the following operations:

1️⃣ **Add a new patient at the beginning**

* Used for critical ambulance arrivals → treated first
* *(Insert from beginning)*

2️⃣ **Add a new patient at the end**

* Used for normal walk-in patients
* *(Insert from end)*

3️⃣ **Add a new patient at a specific position**

* Insert at exact position k (1-based indexing)
* Position Rules:
  + Position 1 → treat immediately
  + Position 2 → treat after first patient
* Example: Insert at position 3 → after 2nd, before 3rd
* *(Insert at specific position)*

4️⃣ **Remove a patient from the beginning**

* First patient has been treated and leaves
* *(Delete from beginning)*

**Q1 — Implementation / Logic**

Write logic in **C++** or detailed **pseudocode** for:

* insertAtBeginning(patientID)
* insertAtEnd(patientID)
* insertAtPosition(patientID, position)
* deleteFromBeginning()

Must include updates to:

* head
* tail
* prev and next pointers

✅ Handle edge cases:

* Empty list insertion
* Deleting when list contains one node
* Inserting at position 1
* Inserting beyond list length (state your handling approach)

**Notes on edge cases handled**

* Inserting into an **empty list** correctly sets both head and tail.
* **Deleting** when only one node exists resets head = tail = nullptr.
* Inserting at **position 1** routes to insertAtBeginning.
* Inserting at a **position > current length + 1** appends at end (clearly documented in comment). If your instructor prefers “reject with error,” that’s a 1-line change.

**Q2 — Dry Run / Trace**

Start with an **empty** ER queue.

Perform each operation and **draw the doubly linked list after every step**:

1. insertAtEnd(101)
2. insertAtEnd(102)
3. insertAtBeginning(200) *(critical patient)*
4. insertAtPosition(150, 2)
5. deleteFromBeginning()
6. insertAtEnd(300)

Then answer:

1. PatientID at **head** = ?  
   b) PatientID at **tail** = ?  
   c) Full **forward traversal** (head → tail)  
   d) Full **backward traversal** (tail → head)

Step 1: insertAtEnd(101)

**Step 1: insertAtEnd(101)**

List:  
[101]  
Head = 101, Tail = 101

**Step 2: insertAtEnd(102)**

List:  
[101] <-> [102]  
(101.next → 102, 102.prev → 101)  
Head = 101, Tail = 102

**Step 3: insertAtBeginning(200) // critical patient**

List:  
[200] <-> [101] <-> [102]  
(200.next → 101, 101.prev → 200)  
Head = 200, Tail = 102

**Step 4: insertAtPosition(150, 2) // after 1st, before previous 2nd**

List:  
[200] <-> [150] <-> [101] <-> [102]  
(200.next → 150 → 101 → 102, and matching prev links)  
Head = 200, Tail = 102

**Step 5: deleteFromBeginning()**

Delete head (200). New list:  
[150] <-> [101] <-> [102]  
Head = 150, Tail = 102

**Step 6: insertAtEnd(300)**

List:  
[150] <-> [101] <-> [102] <-> [300]  
Head = 150, Tail = 300

**Answers after Step 6**

* (a) Head patientID = **150**
* (b) Tail patientID = **300**
* (c) Forward (head → tail) = **150, 101, 102, 300**
* (d) Backward (tail → head) = **300, 102, 101, 150**

**Q3 — Poster Design Requirements**

Your poster must include:

1️⃣ **Title**  
2️⃣ **Sub-Title**

* Student Name
* Roll Number

3️⃣ **Problem Statement**  
4️⃣ **Proposed Solution**  
5️⃣ **Graphical Representation** after each step:

* insertAtEnd(101)
* insertAtEnd(102)
* insertAtBeginning(200)
* insertAtPosition(150, 2)
* deleteFromBeginning()
* insertAtEnd(300)

**✅ GitHub Submission Checklist**

Upload the following:

1. Complete C++ Code
2. Poster (editable format)
3. Poster in **PDF** form



