

EMERGENCY ROOM QUEUE MANAGEMENT SYSTEM

NAME : Hafsa Shawana Noor

1 PROBLEM STATEMENT

Emergency rooms must prioritize patients based on urgency. Critical patients need immediate care and should be placed at the front of the queue, while normal patients wait at the back. This project simulates a priority-based patient queue using a doubly linked list in C++, allowing insertion, deletion, and real-time status display

4 PSEUDOCODE

STRUCT PatientNode
INT a prev, next

CLASS ERQueue
POINTER front, back
INT count

METHOD InsertAtBeginning(id)
Create node → link before front

METHOD InsertAtEnd(id)
Create node → link after back

METHOD InsertAtPosition(id, pos)
Traverse to pos-1 → insert node

METHOD DeleteFromBeginning()
Remove front node

METHOD displayQueue()
Traverse front → back

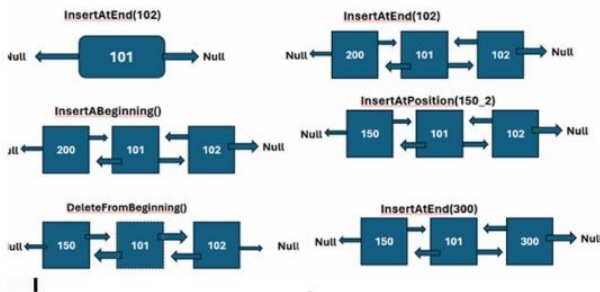
METHOD displayReverseQueue()
Traverse back → front

METHOD displayStatus()
Show front, back, count

2 KEY FEATURE

- InsertAtBeginning(int id) – Adds a critical patient to the front
- InsertAtEnd(int id) – Adds a normal patient to the back
- InsertAtPosition(int id, int pos) – Inserts a patient at a specific position
- DeleteFromBeginning() – Treats and removes the first patient
- displayQueue() – Shows queue from front to back
- displayReverseQueue() – Shows queue from back to front
- displayStatus() – Displays current queue status

5



3 TCONCLUSION

This project successfully simulates a real-world Emergency Room queue using a doubly linked list in C++. It prioritizes critical patients, manages dynamic insertions and deletions, and provides clear visualizations of the queue. Through this system, we demonstrate efficient patient handling, practical use of data structures, and real-time status tracking – all essential for responsive healthcare management.

DATE: _____ DAY: _____
Name: Hafsa Shawana Noor
Title: Emergency Room Queue Management System

1- Inserted At End (101)
Head → [101] ← Tail
prev = null ← [101] → (next = null)
Representation:

HEAD : Patient ID = 101,
Prev = NULL, Next = NULL

2- Insert At End (102)
Head → [101] ← [102] ← Tail
[101] · prev = NULL [101] · Next → [102]
[102] · prev → [101], [102] · next = NULL

3- Insert At Beginning (200)
HEAD → [200] ← [101] ← [102] ← Tail
[200] · prev = NULL [200] · next → [101]
[101] · prev → [200], [101] · next → [102]

Insert At Position (150, 2)
Position is 2 (1-based), so insert 1st node after 200, before 2nd node.
Head → [200] ← [150] ← [101] ← Tail.

Pointers Updated:

• New [150] · prev → [200]
• new [150] · next → [101]
• [200] · next → [150]
• [101] · prev → [150]

DATE: _____ DAY: _____
5- Deleting From Beginning ()

Remove head = node (200)
Head → [150] ← [101] ← [102] ← Tail
Pointers:
• new head is [150], [150] · prev = NULL

Insert At End (300)
Head → [150] ← [101] ← [102] → [300]
Final pointers consistent
• head = node with patient ID (prev = NULL)
• tail = node with patient ID (next = NULL)

After 6 steps Answer:

- (a) Head patient ID : (150)
- (b) Tail Patient ID : 300
- (c) Forward Transversal (head → tail) 150 → 101 → 102 → 300
- (d) Backward Transversal (tail → head) 300 → 102 → 101 → 150