Practical Implementation

Fall 2025

Data Structures (Lab)

Submitted By: Muhammad Shehzad

Section: A

On my honor, as a student of Institute of Management Sciences,
I have neither given not received unauthorized assistance on this
Academic work."

Submitted To: Sir Shayan Shah

Department of Artificial Intelligence,

Institute of Management Sciences, Peshawar.

-:DRY RUN / TRACE:-

OUTPUT:-

InsertionAtEnd(101);
This function insert the valueat end of the node.
For example: head -> NULL tail -> NULL
As we have started from inserting at the end , so the list is empty
After inserting:
Head -> [101] tail -> [101]

- InsertionAtEnd(102);

Again calling this function, will append the 1st value.
for example;
Head -> [101] tail -> [102]
This value has append the 1st value but that value is not deleted.

- insertionAtBeginning(200);

This function insert the value at very beginning means in the first node.
For example: NULL <- [200] <-> [101] <-> [102] -> NULL
So , head = 200 and tail = 102

- insertAtPosition(150 , 2);

This function inserts the value at a specific position.
For example: NULL <- [200] <-> [150] <-> [101] <-> [102] -> NULL
But head and tail remains same
As ; head = 200 and tail = 102

- deleteFromBeginning();

This function deletes the 1st node and the head pointer moves forward.(to the next node).

For example:

Before deletion:

NULL <- [200] <-> [150] <-> [101] <-> [102] -> NULL

After deleting:

NULL <- [150] <-> [101] <-> [102] -> NULL

So, head=150 and tail = 102

- insertAtEnd(300);

This function will insert the value at the end of the list.

For example:

NULL <- [150] <-> [101] <-> [102] <-> [300] -> NULL

Head =150 and tail = 300

The value of the tail has been changed because of insertion at the end.

-:ANSWERS:-

1) The patient id at the head is 150

2) The patient id the tail is 300.

3) After calling display function the list will be printed as:

150 – 101 – 102 – 300

4) Printing backward:

300 – 102 – 101 – 150

---

im | sciences

Emergency Room Queue Management Using Doubly Linked List

Sub Title:

M.Shehzad
BS(AI) 'A'

DSA C++

problem Statement:
In a hospital emergency room, patients must be treated based on priority.
Critical patients (arriving by ambulance) are treated first, while normal patients can wait.
Because patient arrivals and discharges change quickly, the hospital needs a flexible system to manage the ER queue dynamically.

Proposed Solution:

We implement the ER Queue using a Doubly Linked List in C++. Each patient is a node that stores:

- patientID
- Pointer to the previous patient
- Pointer to the next patient

Graphical Representation

Step 1: [101]head → [101] ← tail
Step 2: head → [101] <-> [102] ← tail
Step 3: head → [200] <-> [101] <-> [102] ← tail
Step 4: head → [200] <-> [150] <-> [101] <-> [102] ← tail
Step 5:head → [150] <-> [101] <-> [102] ← tail
Step 6: head → [150] <-> [101] <-> [102] <-> [300] ← tail

Operations:-

insertAtBeginning()
Add critical patient (treated first)
insertAtEnd()
Add normal patient (waits in queue)
insertAtPosition()
Add patient at specific priority level
deleteFromBeginning()

Conclusion:-

- Efficient management of ER patients using Doubly Linked List
- Supports dynamic insertion and deletion
- Keeps track of both head (critical) and tail (normal) patients
- Demonstrates real-world hospital queue management

the project of hoepital emergency room queue