Fall 2025

Data Structure & Algorithms

Submitted By: Abuhuraira

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 Ali Shah
Department of Artificial Intelligence,
Institute of Management Sciences, Peshawar.

Functions and their working:

insertAtBeginning(patientID)

Creates a new node with the given patientID.

If the list is empty, both head and tail point to this new node.

Otherwise, the new node's next points to the current head, and the current head's prev points back to the new node.

Then, head is updated to the new node.

insertAtEnd(**patientID**)

Creates a new node.

If the list is empty, both head and tail are set to the new node.

Otherwise, it attaches the new node at the end by updating the current tail's next pointer and setting the new node's prev pointer to the old tail.

Finally, tail is updated to this new node.

insertAtPosition(patientID, position)

If the position is 1 or the list is empty, it calls insertAtBeginning.

Otherwise, it traverses the list to find the position (or the end if the position is larger than the current length).

If the position is beyond the end, it calls insertAtEnd.

Otherwise, it links the new node between the correct nodes by updating next and prev pointers of neighboring nodes.

deleteFromBeginning()

If the list is empty, it prints "Queue empty."

If there's only one node, it deletes it and makes the list empty.

Otherwise, it deletes the head node, updates head to the next node, and sets head->prev to nullptr.

All functions properly update:

head and tail pointers

prev and next connections

Edge cases handled:

Empty list insertions/deletions

Single node deletions

Insertions at position 1

Insertions at position greater than length

Q2. Dry Run / Trace

Start: Empty list

Step Operation Forward Traversal (head → tail) Backward Traversal (tail → head)

1 insertAtEnd(101) 101 101

2 insertAtEnd(102) 101 <-> 102 102 <-> 101

3 insertAtBeginning(200) 200 <-> 101 <-> 102 102 <-> 101 <-> 200

4 insertAtPosition(150, 2) 200 <-> 150 <-> 101 <-> 102 102 <-> 101 <-> 150 <-> 200

5 deleteFromBeginning() 150 <-> 101 <-> 102 102 <-> 101 <-> 150

6 insertAtEnd(300) 150 <-> 101 <-> 102 <-> 300 300 <-> 102 <-> 101 <-> 150

After Step 6:

(a) Head patientID: 150

(b) Tail patientID: 300

(c) Forward traversal: 150 → 101 → 102 → 300

(d) Backward traversal: 300 → 102 → 101 → 150

im | sciences

Poster Designed: