→ ENQUEUE in a queue linked list:

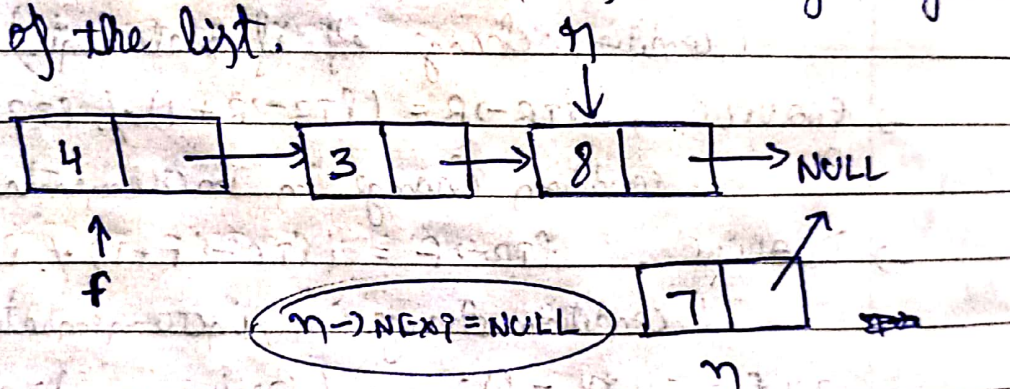
→ Enqueue in a queue linked list is very much similar to just inserting at the end in a linked list. If we discussed this thoroughly in our past lectures, you should not find this difficult. Inserting a new node at the end requires you to follow few steps:

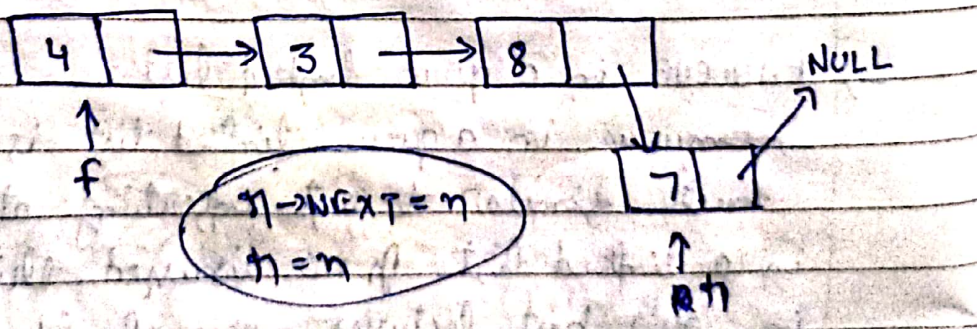
1) Check if there is a space left in the heap for a new node.

2) If there is, create a new node n, assign its memory in heap, and fill in its data with the new value the user has given.

3) Point the next member of this new node n to NULL, and point the next member of the rr to n. And make rr equal to n. And we are done.

4) There is one exception here. When we insert the first element, both f and rr are pointing to NULL. So, instead of just making rr equal to n, we make f equal to n as well. This marks the beginning of the list.





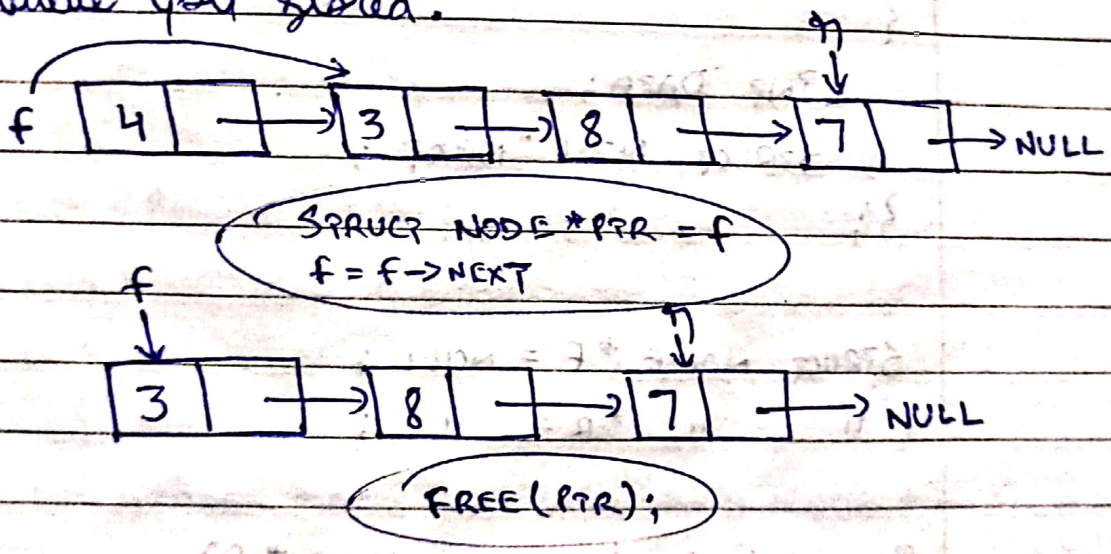
→ Dequeue in a queue linked list:

→ Dequeuing in a queue linked list is very much similar to deleting the head node in a linked list. Deleting the head node

Page No.
Date

from the list requires you to follow these 3 steps:

1) Check if the queue list is already not empty using the ISEMPTY function.
2) If it is, RETURN -1. Else create a new node PTR and make it equal to the f node. And don't forget to store the data of the f node in some integer variable.
3) Make the f equal to the next member of f, and free the node PTR. Return the value you stored.



→ Condition for ISEMPTY:
→ The only condition for the queue linked list to the empty is that the f node is NULL, which means there is no beginning, hence no element.

→ Condition for ISFULL:
→ Queues implemented using linked lists never usually become full until the space in the heap memory is exhausted.

Therefore, the only condition for the queue linked list to be full is that the new node becomes NULL when created.