Lab 3

In this lab, we will implement queue using linked list. A brief idea is available in the Queue slide. Now, we will write the code in the lab. Understnading and writing this code might give you a better idea on solving your programming assignment 2. You do not have to submit this lab's code.

♦ <u>Problem: Implement a queue using linked list :</u>

- <u>The main concept:</u> anything added to the linked list will be inserted at the end and anything deleted should be deleted from the front of the linked list.
- But, to add anyting at the end, traversing to the end everytime is expensive/time consuming. Imagine havig a queue of size 1000000. For inserting next item, you will need to visit all 1000000 items to find the last node in the queue.

So, we would like to avoid such traversal everytime we insert a new item. To solve the problem we will maintain two pointers:

- One for front of the list
- One for the back

- Our struct to store the queue, would actually store two pointers to linked list structs:

- The first one would point to the head of the list.
- The second one would always point to the last node in that list

See the node structure and queue structure bellow:

```
// Stores one node of the linked list.
struct node {
    int data;
    struct node* next;
};
```

```
// Stores our queue.
struct queue {
    struct node* front;
    struct node* back;
};
```

<u>Functions needed:</u>

I. **<u>init</u>**: This function should make front and rear of the queue as NULL.

II. **enqueue**:
   a) Create a new node and store the inserted value into it.
   b) Link the back node's next pointer to this new node.
   c) Move the back node to point to the newly added node.

III. **dequeue**:
   a) Store a temporary pointer to the beginning of the list.
   b) Move the front pointer to the next node in the list.
   c) Free the memory pointed to by the temporary pointer.

IV. **front**:
Directly access the data stored in the first node through the front pointer to the list.

V. **empty**:
Check if both pointers (front, back) are null.