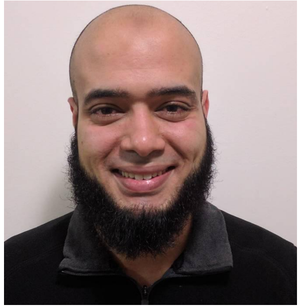
# Data Structures

# Linked-list-based Queue

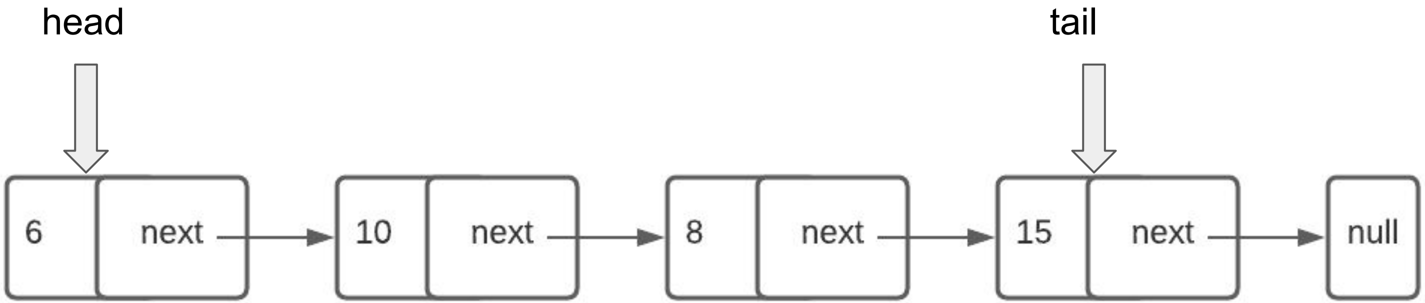
**Mostafa S. Ibrahim**
*Teaching, Training and Coaching since more than a decade!*

*Artificial Intelligence & Computer Vision Researcher*
*PhD* from Simon Fraser University - Canada
*Bachelor / Msc* from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)



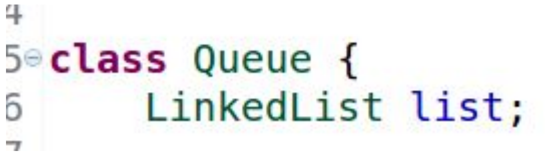
# Similar to Linked-list-based Stack

- Clearly, we can use a linked list to solve fixed memory issue
  - Downside: clearing a queue is O(n) using linked list
- Head is actually our front and tail is our rear
  - All what we need is to add in tail and get from head!



# Aggregation

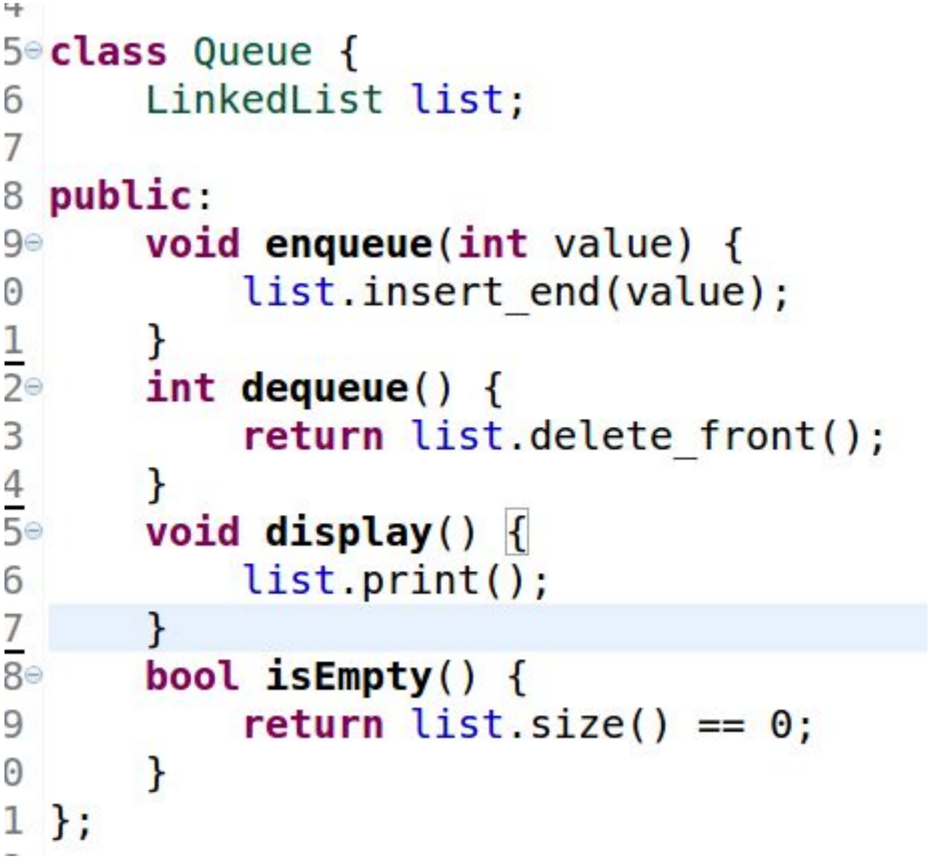
- We can get our lecture code for Singly linked-lists and do renamings
- In OOP/SWE, aggregation/composition is to let your class use objects of other classes
- Many Data structures ADT can be implemented in terms of other data structures
    - We can implement a stack using a queue
    - We can implement a queue using a stack
    - And so on, regardless of complexity



```
class Queue {
    LinkedList list;
```

# Queue

- We will extend our old linked list to have 2 member functions
  - Delete front (act as dequeue)
  - size() to # of elements
- Now Queue, can just **delegate** the class to the linked list
- In homework, there are some more exercise of this style
  - E.g. a queue using 2 stacks



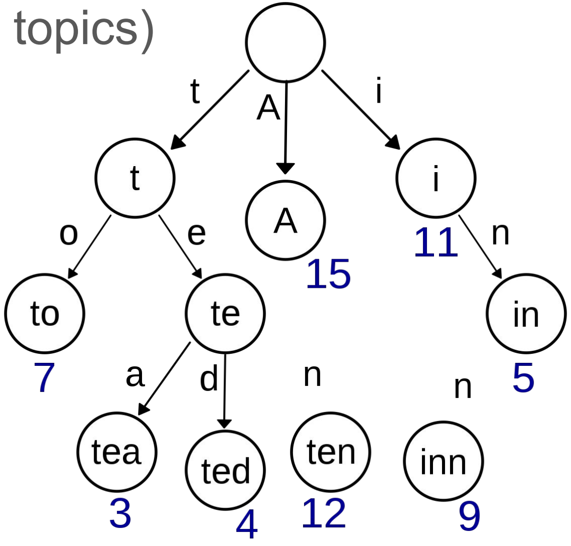
```
class Queue {
    LinkedList list;

public:
    void enqueue(int value) {
        list.insert_end(value);
    }
    int dequeue() {
        return list.delete_front();
    }
    void display() {
        list.print();
    }
    bool isEmpty() {
        return list.size() == 0;
    }
};
```

# Tips

- Stack: Follows FILO
  - It helps reversing data
  - It involves some recursive nature: (()()(()))
- Queue: Follows FIFO
  - It can be used to simulate queues (restaurant, hospital, customer calls)
- **Lessons from homework (to keep in mind)**
  - We can use an available data structure to implement another (aggregation)
  - If you moved data from Queue to Stack, then back to Queue: Now Queue is **reversed**
  - If Queue has N elements, pushing an element then enqueue/Dequeue n times add it at **front**
    - [1, 2, 3, 4] ⇒ insert front (5) ⇒ [1, 2, 3, 4, 5] ⇒ [5, 1, 2, 3, 4]
  - It can be used to act as a **sliding window** over data: [1, 2, 3, 4, 5, 6, 7, 8]
    - window=3: [1, 2, 3], [2, 3, 4], [3, 4, 5], [4, 5, 6], … etc

# Linear and Non-linear Data Structures

- **Linear** data structure:  elements arranged & connected in **sequential** manner
  - Array, Vector, Linked List Stack and Queue are linear data structures
- A **non-linear** data structure: elements can have **multiple paths** to connect to other elements. Example such as Tree / Trie (in next topics)
  - On right nodes are linked to form a tree



Img src

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."