Queue

Queue data structure facilitates First In First Out (FIFO) relation for the data access. It can be implemented in two ways:

- Array based implementation
- Linked List based implementation

One might ask, why not simply use Stack? The reason is that items might get buried in stacks and not appear in the top for long time. Stack is unfair to old items! The table belows shows complexities of different operations for array based Queue implementation.

Operation	Description	Complexity
$enqueue$	Insert at the top	$O(1)$
$dequeue$	Remove the bottom item	$O(N)$
is_empty	Return True if Queue is empty, else False	$O(1)$
$make\_empty$	Empty the Queue	$O(1)$

Since removing an element from top of the list requires shifting all the remaining items forward, $dequeue$ operation takes $O(N)$. For $O(1)$ $enqueue$ and $dequeue$, we can use two index variables $first$ and $last$.

- $Enqueue\ x$: increment $last$ and set $list[last] = x$
- $Dequeue$: return $list[first]$ and increment $first$

First ↓ Last ↓

0	1	2	3	...	N-2	N-1				MAX
A	B	C	D	---	K	L				

Furthermore, since each $dequeue$ generates an available space, we can circularly link $last$ and $first$ variable. If $first/last$ has reached end of the list and there is space available at the front, wrap them to 0.