**March 17, 2021**

1. **Stack (review)**

* **Applications: calling stack for recursion, transformation stack, for compilation**
* **Assumption: we are allowed to push or pop an element only at the top of the stack.**
* **Characteristic: FILO (First In Last Out)**
* **Member Functions: Push( ) and Pop( )**

1. **Queue**

**Applications: waiting queue for processes on a computer, a queue in simulation**

**Characteristic: FIFO (First In First Out)**

**Implementation: static array, dynamic array, or linked list (head🡪 front, tail 🡪 rear)**

1. **Array Implementation**

**A.1 simple implementation (drawback: too many shifting operations)**

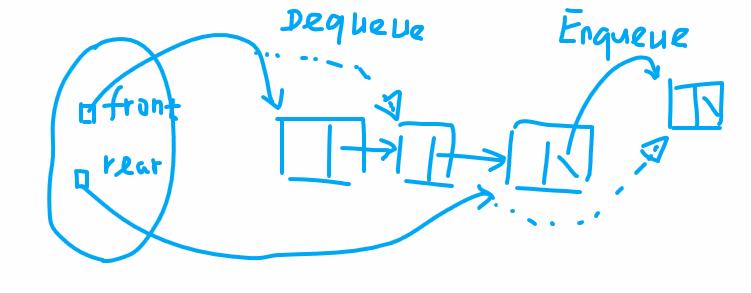
**A.2 second implementation (introducing two data members: front and rear; drawback: one exception case)**

**A.3 third implementation (introducing another data member: length of the queue or introducing a reserved element)**

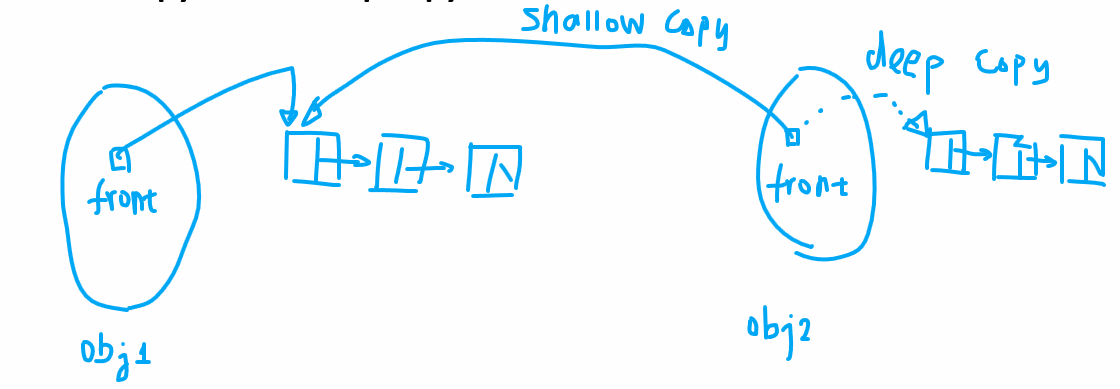
**We put the reserved element in front of the first element of the queue; consider front = the location of the reserved element.**

1. **Implementation by Linked List**

**Use two pointer variables: front and rear**

****

**Shallow copy versus deep copy:**

****

**Note that the deep copy is needed only when a data member is a linked list.**

**Deep copy: It is basically a while loop that traverses the old linked list and during each step of the traversal, we create a new node and copy the value from the old linked list to the new linked list. Repeat the process till the end of the old linked list and at the last step, set the next field of the last node of the new linked list to NULL. We rely on two pointers: p1 and p2; the first one points to the old linked list and the second one points to the new linked list. Don’t use topPtr in the while because it changes its value and we will lose the track for the beginning memory address of the linked list. You should always use p1 and p2 (temporary pointer variables in the while loop).**