Practice Questions Queue

Question 1

Implement a **MyQueue** class that implements a queue using two stacks which should have ***enQueue*** and ***deQueue*** and ISEMPTY and ISFULL functions.

Hint:

enQueue(q, x):

- *While stack1 is not empty, push everything from stack1 to stack2.*
- *Push x to stack1 (assuming size of stacks is unlimited).*
- *Push everything back to stack1.*
- *Here time complexity will be O(n)*

deQueue(q):

- *If stack1 is empty then the error*
- *Pop an item from stack1 and return it*
- *Here time complexity will be O(1)*

Question 2

Create a class Deque (Doubly Ended Queue)

Deque will have following variables

T * Data // Templated Pointer

int Front

int Rear

int Size, Capacity;

Add these Following functions:

Deque () // Set the default values

Deque (int Size) // create a fixed-size array at the start that can be grown in future & Set the default values

~Deque () // delete the memory allocated

bool insertFront(T Value)// Adds an item at the front of Deque. If the Queue Is full it should regrow itself by a factor of 2.

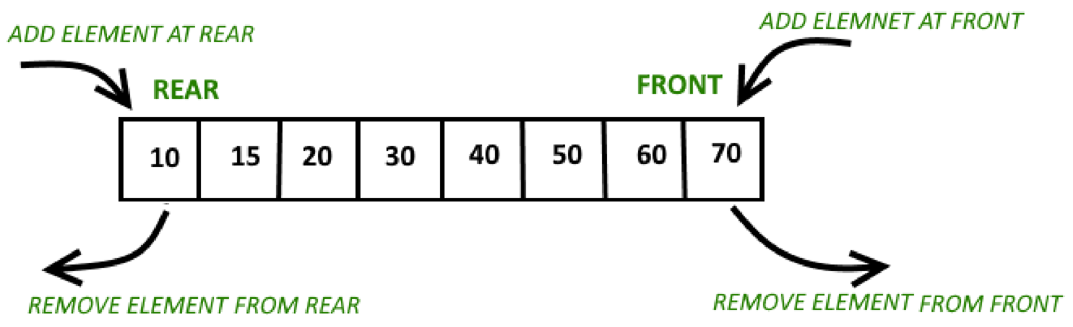
bool insertLast(T Value)// Adds an item at the rear of Deque. If the Queue Is full it should regrow itself by a factor of 2.

bool deleteFront(T& ReturnValue)// Deletes an item from front of Deque. Handle the case of empty Deque.

bool deleteLast(T& ReturnValue)// Deletes an item from the rear of Deque. Handle the case of empty Deque also.

bool IsEmpty() // (Size == 0)

bool IsFull() //(Size == Capacity)



Question 3

Implement a ***MinMaxQueue*** class which should have *enQueue* and *deQueue* and getMax and getMin functions.

enQueue

- Insert the element into the queue structure.
- If the size of the **Deque structure (implemented in Question 2)** is empty that is the size of the Deque is 0. Then, Insert the element from the **back**.
- Otherwise, If there are some elements in the Deque structure then pop the elements out from the **Deque** until the back of the Deque is greater than the current element and then finally insert the element from the back.

deQueue

- If the first element of the Deque is equal to the front element of the queue then pop the elements out from the Queue and the Deque at the same time.
- Otherwise, Pop the element from the front of the queue to maintain the order of the elements.

Get Minimum

Return the front element of the Deque to get the minimum element of the current element of the queue.