Agenda

1) Introduction to queue
2) Queue functions in Java
3) Reverse first K elements of given queue
4) Create N no. using only 1, 2 and 3
5) Adapter (Queue using Stack)

what is queue

queue follows FIFO

↳ first in first out

Real life examples

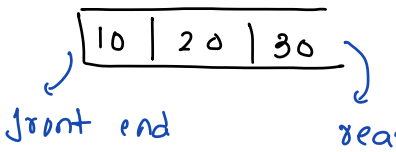
→ task scheduling (printing)

How to create and use a queue in Java

```
Queue<Integer> q = new ArrayDeque<>();
```

q → name of variable

```
q.add(10);
q.add(20);
q.add(30);
sopln(q.remove()); ⟶ 10
sopln(q.peek()); ⟶ 20
```



q.add(x) : x will get added at rear end.

q.remove() : removal will from the front end.

q.add(x), q.remove(), q.peek() → O(1)

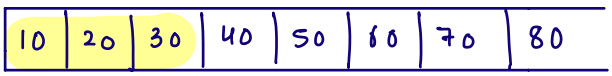
Why not AL for FIFO feature

removal of 0^{th} index ele in AL is an O(n) opr.

list.remove(0) ⟶ O(n)

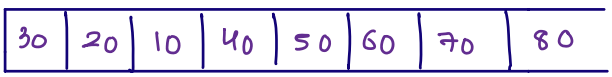
Q.1 Given a queue, reverse first K elements of it.

q :



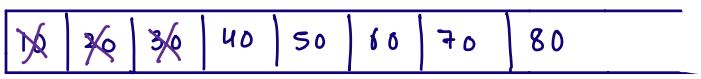
K = 3

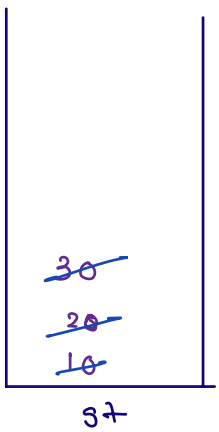
after reversal of first K elements



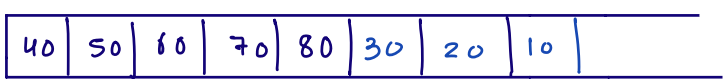
Expected TC: O(n)

step1 : remove K elements from q and push them to stack





step 2 : pop the entire content from stack and it to queue

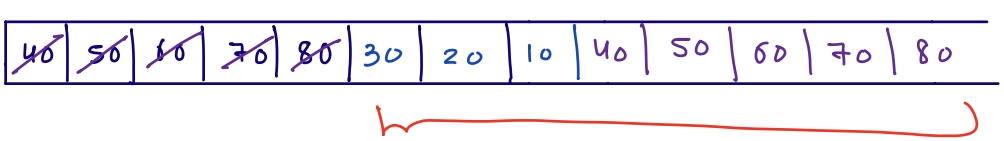


TC: O(n)

SC: O(K)

step 3 : remove n-K ele from q and add them q.

n = 8
K = 3
n-K = 5



Q.2 Create N no. in Ascending order using only 1, 2 & 3 as digits and return these numbers.

N = 4 ans: 1 2 3 11

N = 7 ans: 1 2 3 11 12 13 21

N = 10 ans: 1 2 3 11 12 13 21 22 23 31

N = 13 ans: 1 2 3 11 12 13 21 22 23 31 32 33 111

~~1~~ ~~2~~ ~~3~~ ~~11~~ ~~12~~ ~~13~~ ~~21~~ ~~22~~ ~~23~~ ~~31~~ ~~32~~ ~~33~~

111 112 113 121 122 123 131 132 133 211 212 213

221 222 223 231 232 233 311 312 313 321 322 323

331 332 333

queue → generating no.

arrayList → to store ans

N = 8

q: | 1 | 2 | 3 | 11 | 12 | 13 | 21 | 22 | 23 |

ans: (AL) | 1 | 2 | 3 | 11 | 12 | 13 | 21 | 22 |

count = 8 9

↓

no. generated so far

```
q.add(1); q.add(2); q.add(3);
count = 3;
while(ans.size() < N) {
    int temp = q.remove();
    ans.add(temp);
    if(count < N) {
        //generate no. using temp
        int v1 = temp*10 + 1;
        int v2 = temp*10 + 2;
        int v3 = temp*10 + 3;
        q.add(v1); q.add(v2); q.add(v3);
        count += 3;
    }
}
```

N = 14

q : | 1 | 2 | 3 | 11 | 12 | 13 | 21 | 22 | 23 | 31 | 32 | 33 | 111 | 112 | 113 |

ans: | 1 | 2 | 3 | 11 | 12 | 13 | 21 | 22 | 23 | 31 | 32 | 33 | 111 | 112 |

```
q.add(1); q.add(2); q.add(3);
count = 3;
while(ans.size() < N) {
    int temp = q.remove();
    ans.add(temp);
    if(count < N) {
        //generate no. using temp
        int v1 = temp*10 + 1;
        int v2 = temp*10 + 2;
        int v3 = temp*10 + 3;
        q.add(v1); q.add(v2); q.add(v3);
        count += 3;
    }
}
```

count = 3 6 9 12 15

Adapter

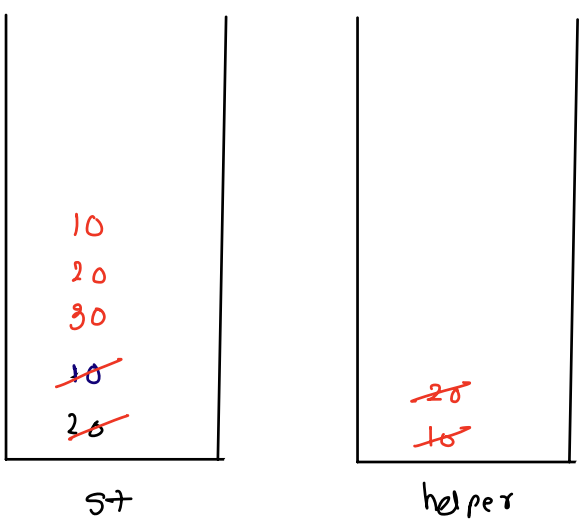
Q. Implement Queue functions using Stack (remove efficient)

↳ (as data member)

feature of queue → maintaining FIFO

```
class Adapter {

    void add (int x)

    int remove ()      ⟶  O(1)

    int peek ()        ⟶  O(1)

}
```

```
Adapter q = new Adapter();

q.add (10)
q.add (20)
q.add (30)
q.remove() → O(1)
```



add(x) → i) Shift content from st to helper

ii) push x to st

iii) shift content from helper to st

remove() → st.pop()

peek() → st.peek()

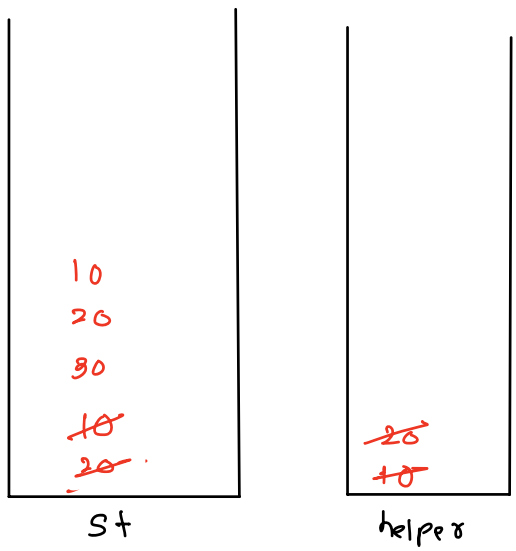
Adapter q = new Adapter();

q.add(10)

q.add(20)

q.add(30)

q.remove()



```java
public static class UserQueue {
    /** Initialize your data structure here. */

    static Stack<Integer>st = new Stack<>();

    UserQueue() {

    }

    /** Push element X to the back of queue. */
    static void push(int X) {
        Stack<Integer>helper = new Stack<>();

        //shift content from st to helper
        while(st.size() > 0) {
            helper.push(st.pop());
        }

        //push X to the st
        st.push(X);

        //shift content from helper to st back
        while(helper.size() > 0) {
            st.push(helper.pop());
        }
    }

    /** Removes the element from in front of queue and returns that element. */
    static int pop() {
        return st.pop();
    }

    /** Get the front element of the queue. */
    static int peek() {
        return st.peek();
    }

    /** Returns whether the queue is empty. */
    static boolean empty() {
        return st.size() == 0;
    }
}
```