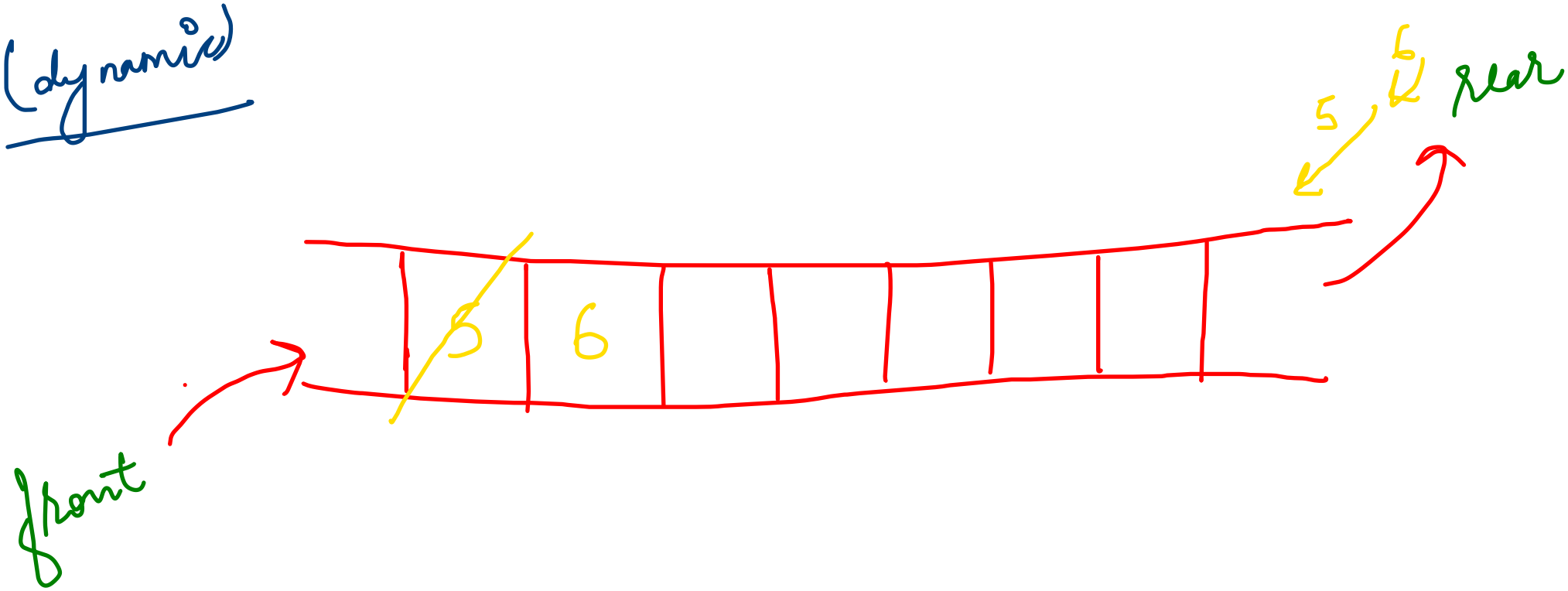
Queue : FIFO (First In First Out)

(dynamic)



add element → rear

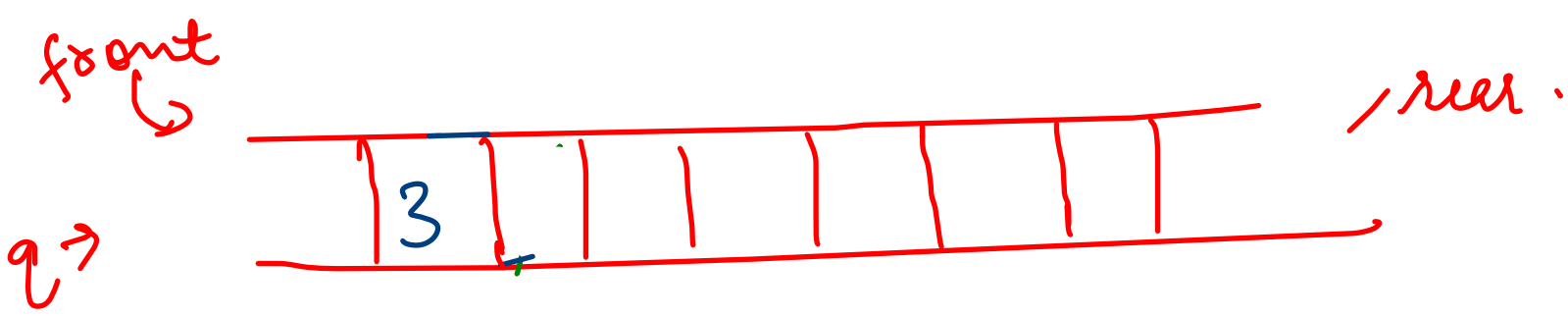
remove element → front

Syntax

```
Queue<DataType> que = new LinkedList<>();
```

In-built

```
→ que.add(value);   // add elements from rear end
→ que.remove();     // remove from front & print it
→ que.poll();       // remove from front & print it
→ que.peek();       // return front element w/o removing.
→ que.size();       // return size of Queue
→ que.isEmpty();    // whether Queue empty or not.
```




```
q.add(5);
q.add(6);
q.peek(); //5
q.poll(); //5
q.peek(); //6
q.isEmpty(); //false
q.size(); //1
```

```
q.poll(); //6
q.add(3);
```

HW_Queue Reversal 5

```
Scanner s = new Scanner(System.in);
int n = s.nextInt();

int que[] = new int[n];
for(int i = 0; i < n; i++){
    que[i] = s.nextInt();
}

Stack<Integer> st = new Stack<>();
for(int i = 0; i < n; i++){
    st.push(que[i]);
}
    while(!st.isEmpty()){
        System.out.print(st.pop() + " ");
    }
}
```

q:- 2, 3, 2

~~A-2~~

~~B-3~~

~~C-2~~

A-1

B-2

C-1

~~A-0~~

~~B-2~~

~~C-0~~

B-1

~~B-1~~

~~1+1+1+1+1+1~~ = 6 sec.

HW_Longest K unique characters substring

```java
 public static int longestKUniqueSubstring(String s, int k) {
if (s == null || s.length() == 0 || k <= 0) {
    return -1;
}

HashMap<Character, Integer> charCountMap = new HashMap<>();
int maxLength = -1;
int left = 0;

for (int right = 0; right < s.length(); right++) {
    char rightChar = s.charAt(right);
    charCountMap.put(rightChar, charCountMap.getOrDefault(rightChar, 0) + 1);

    while (charCountMap.size() > k) {
        char leftChar = s.charAt(left);
        charCountMap.put(leftChar, charCountMap.get(leftChar) - 1);
        if (charCountMap.get(leftChar) == 0) {
            charCountMap.remove(leftChar);
        }
        left++;
    }

    if (charCountMap.size() == k) {
        maxLength = Math.max(maxLength, right - left + 1);
    }
}

return maxLength;
```