Queue Syntax Learning

1. Declare an Empty $queue\ s$.

2. Take Single Integer T as input.

3. For next T Lines format $(case, x(optional))$

- case 1. $Print$ the $size$ of the $queue$ in a separate line.
- case 2. $Remove$ an element from the queue. If the queue is empty then print -1 in a separate line.
- case 3. Add Integer x to the $queue\ s$.
- case 4. $Print$ an element at the $front$ of the $queue$. If queue is empty print -1 in a seperate line.

	ans
5	
1	0
2	-1
3 9 x	9
4	1
1	

dry run

queue

x = 9

rear

front

9

T = 5

Code

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    Queue<Integer> que = new LinkedList<>();
    int t = scn.nextInt();
    while (t-- > 0) {
        int n = scn.nextInt();
        if ( n == 1 ) {
            System.out.println( que.size() );
        } else if ( n == 2 ) {
            if ( que.size() == 0 ) {
                System.out.println("-1");
            } else {
                que.poll();
            }
        } else if ( n == 3 ) {
            int x = scn.nextInt();
            que.add(x);
        } else if ( n == 4 ) {
            if ( que.size() == 0 ) {
                System.out.println("-1");
            } else {
                int ans = que.peek();
                System.out.println(ans);
            }
        }
    }
}
```

Print Binary

$n = 12$

0 →	0000
1 →	0001
2 →	0010
3 →	0011
4 →	0100
5 →	0101
6 →	0110
7 →	0111
8 →	1000
9 →	1001
10 →	1010
11 →	1011
12 →	1100

binary (2)

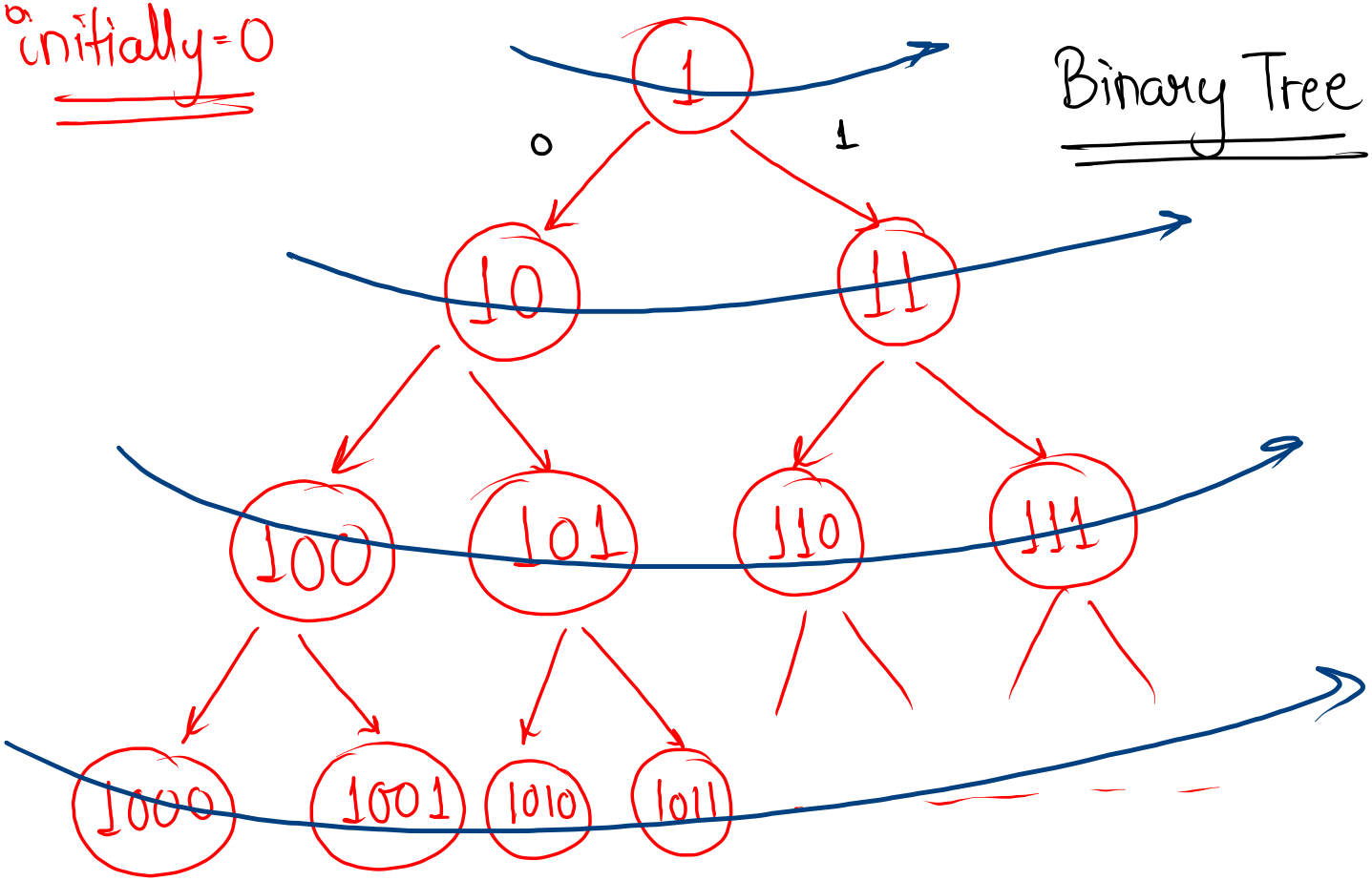
how normal no. are created (decimal)

00	10	20	90
01	11	21	91
02	12	22	92
03	13	23	93
04	14	24	94
05	15	25	95
06	16	26	96
07	17	27	97
08	18	28	98
09	19	29	99

100	110	120	190
101	111	121	191
102	112	122	192
103	113	123	193
104	114	124	194
105	115	125	195
106	116	126	196
107	117	127	197
108	118	128	198
109	119	129	199

initially = 0

Binary Tree



Note:- add 0 and 1 in right side of our current no.

dry run

rear

~~1~~	~~10~~	~~11~~	~~100~~	101	110	111	1000	1001	...

front

n = no.'s that we are removing

ans = 0
1
10
11
100
⋮
until n

rem = 1	10	11	100
left = 10	100	110	1000
right = 11	101	111	1001

left = rem + "0"
right = rem + "1"

Code

T.C

O(n)

S.C

O(n)

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    printBinary(n);
}
public static void printBinary( int n ) {
    Queue<String> que = new LinkedList<>();
    que.add("1");
    for (int i = 0; i < n; i++) {
        String rem = que.poll();
        System.out.print(rem + " ");

        String left = rem + "0";
        que.add(left);

        String right = rem + "1";
        que.add(right);
    }
}
```

Convertion (Extra)

convert binary to decimal

$\Rightarrow$ 1 1 0 1 1 0 0 1

2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0

$\Rightarrow$ sum $= 2^7 + 2^6 + 0 + 2^4 + 2^3 + 0 + 0 + 1$

$\Rightarrow 128 + 64 + 16 + 8 + 1$

$\Rightarrow$ 217

decimal to binary

2	217	1
2	108	0
2	54	0
2	27	1
2	13	1
2	6	0
2	3	1
	1	

First Negative Integer 2 (Imp Hard)

$$\begin{array}{cccccccccc} & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ arr = [& -8, & 2, & 3, & -6, & 10, & 3, & 1, & 4, & -2] \end{array}, \quad K = 3$$

$$ans = [-8, -6, -6, -6, 0, 0, -2]$$

$$T.C = O(n * K)$$ Brute force

psudo code

1) save indexes of -ve element for first k size window.

2) iterate until (n-1)

2.1) que size == 0

curr ans = 0

2.2) else

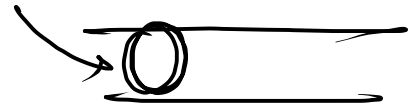
curr ans = peek

2.3) loop $\frac{\text{peek}() < i-k+1}{\text{que.poll}()}$

2.4) que.add(i)

faith:-

que will always be storing indexes of -ve elements for current k size window



Code

T.C = O(n)

S.C = O(n)

```java
public static void firstNegetiveInteger(int[] arr, int n, int k) {
    // que contain index of -ve elements only
    Queue<Integer> que = new LinkedList<>();
    int i = 0;
    // store -ve elements indexes for first window
    while (i < k) {
        if ( arr[i] < 0 ) {
            que.add(i);
        }
        i++;
    }

    while (i < n) {
        // first negetive element of current window
        if ( que.size() > 0 ) {
            System.out.print( arr[que.peek()] + " " );
        } else {
            System.out.print( "0 " );
        }

        // to remove all elements which are our of window
        while ( que.size() > 0 && que.peek() < (i - k + 1) ) {
            que.poll();
        }

        // keep on adding -ve elements
        if ( arr[i] < 0 ) {
            que.add(i);
        }
        i++;
    }

    // for last window
    if ( que.size() > 0 ) {
        System.out.print( arr[que.peek()] + " " );
    } else {
        System.out.print( "0 " );
    }
}
```

i

0	1	2	3	4	5	6	7	8

arr = [-8, 2, 3, -6, 10, 3, 1, 4, -2], k = 3

que

~~0~~	~~3~~	8

ans = [-8, -6, -6, -6, 0, 0, -2]