Max Number of K-Sum Pairs

arr = [4 , 7 , 1 , 3 , 6 , 8 , 5] , k = 10

0 1 2 3 4 5 6

num1

Logic

map array ele → index

4 → 0

7 → 1

1 → 2

3 → 3

6 → 4

8 → 5

5 → 6

num1 + num2 = k

num2 = k - num1

num1 = 4 7 1 3 6 8 5

num2 = 6 3 9 7 4 2 5

Count = 0 1 2

Code

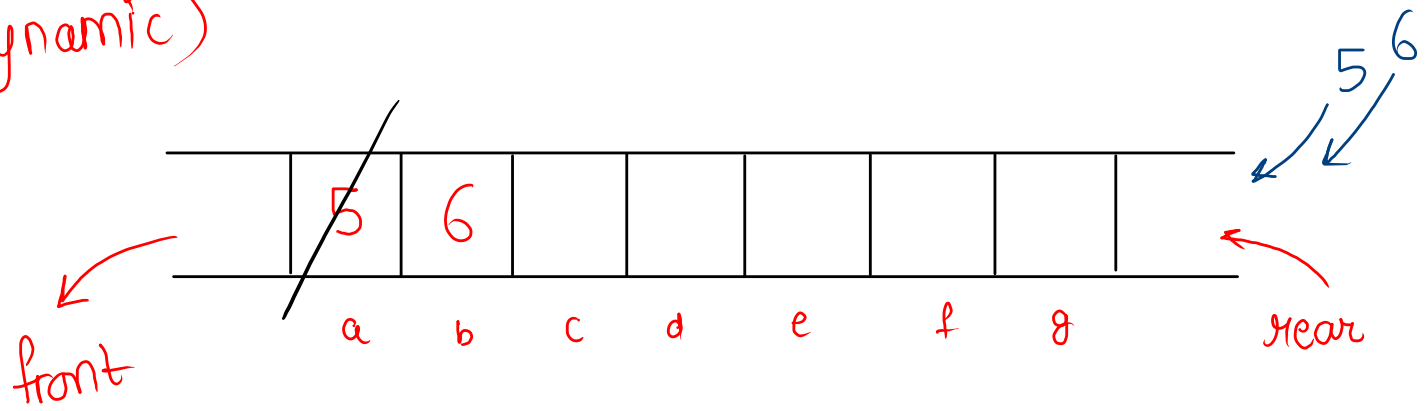
```
public static int maxNumPairs(int[] arr, int n, int k) {

    HashMap<Integer, Integer> map = new HashMap<>();
    for (int i = 0; i < n; i++) {
        map.put( arr[i], i );
    }

    int count = 0;
    for (int i = 0; i < n; i++) {
        int num1 = arr[i];
        int num2 = k - num1;
        if ( map.containsKey(num2) ) {
            if ( i != map.get(num2) ) {
                count++;
                map.remove(num1);
                map.remove(num2);
            }
        }
    }
    return count;
}
```

T.C = O(n)

S.C = O(n)

⟹ Queue [FIFO :- First In First Out]

(dynamic)



Note :- add elements from rear side only
& remove elements from front side only

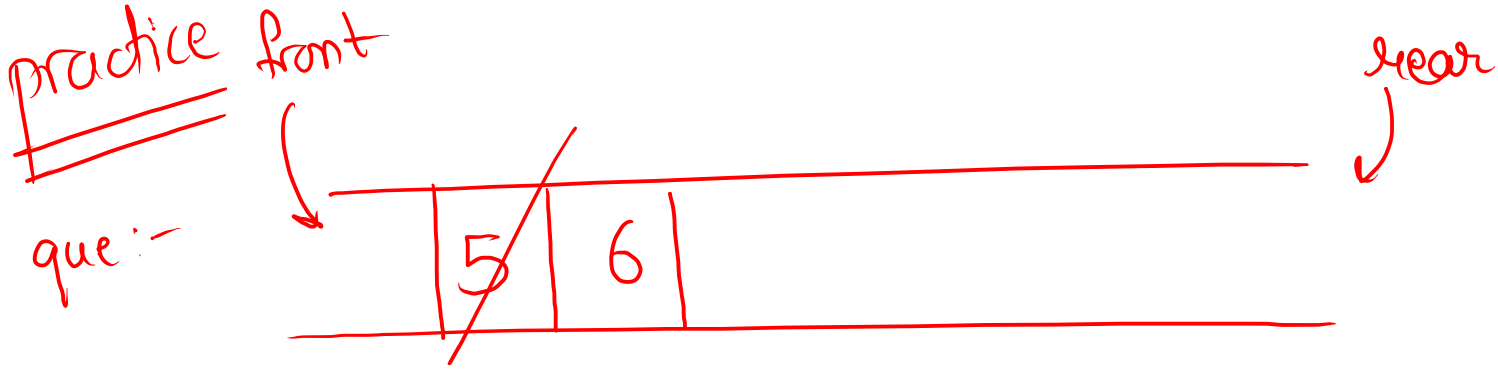
Syntax

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

inbuilt functions

```
que.add(value);   // add element from rear end
que.remove();     // remove front element & return it
que.poll();       // remove front element & return it
que.peek();       // return front element without removing it

que.size();  / que.isEmpty();
```

practice




```
que.add(5);
que.add(6);
que.peek();   // 5
que.poll();   // 5
que.peek();   // 6
que.isEmpty(); // false
```

Queue Syntax Learning

Code

```
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    Queue<Integer> que = new LinkedList<>();
    int t = scn.nextInt();
    for (int i = 0; i < t; i++) {
        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 {
                int val = que.poll();
               // System.out.println(val);
            }
        } 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 val = que.peek();
                System.out.println(val);
            }
        }
    }
}
```

Print Binary

decimal

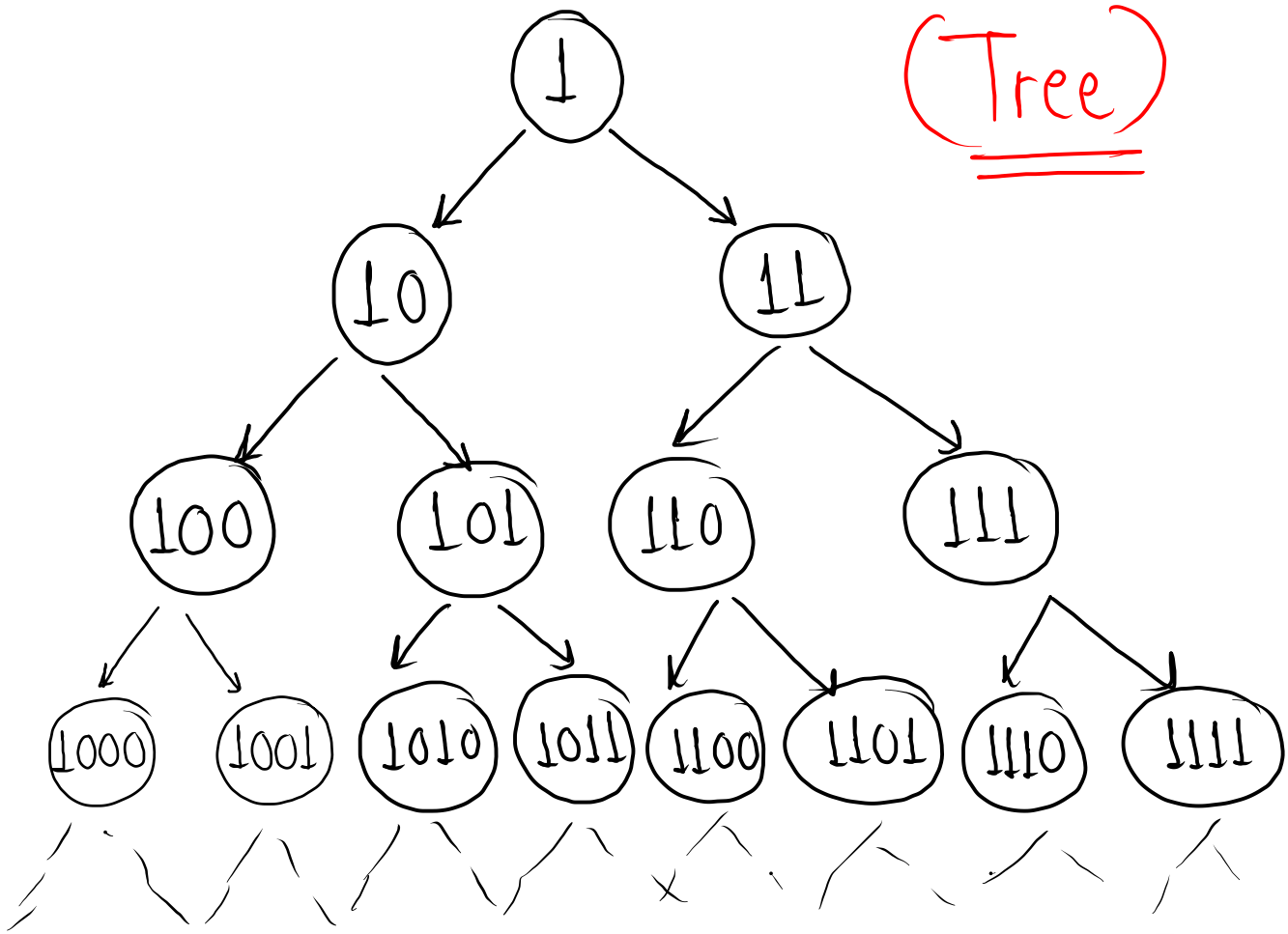
00	10	20	90
01	11	21	91
02	12	22	92
03	13	23	93
04	14	24	94
05	15	25	95
06	16	26	96
07	17	27	97
08	18	28	98
09	19	29	99

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

binary

0000 → 0	1000 → 8
0001 → 1	1001 → 9
0010 → 2	1010 → 16
0011 → 3	1011 → 11
0100 → 4	1100 → 12
0101 → 5	1101 → 13
0110 → 6	1110 → 14
0111 → 7	1111 → 15

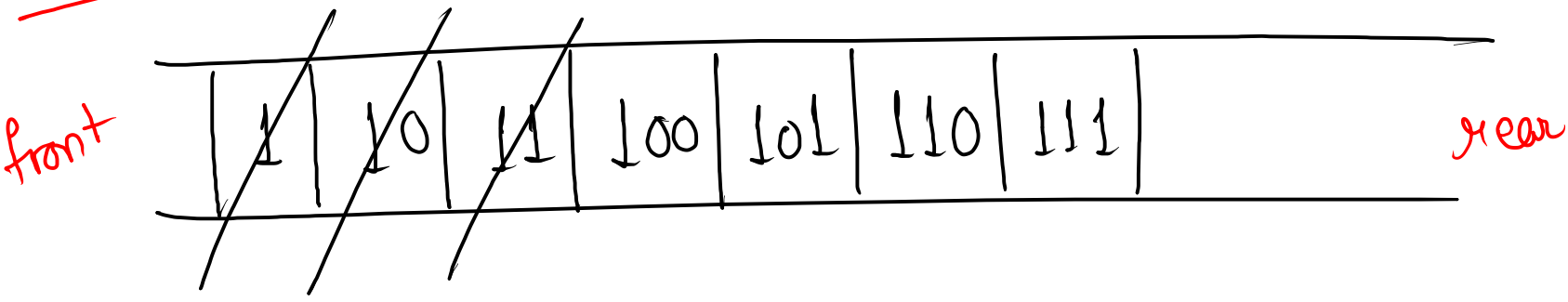
0
1
10
11
100
101
110
111
...



Tree

Note:- add 0 at right side
add 1 at right side

dry run



int rem = que.poll();

concatenate 0 with rem

concatenate 1 with rem

1	10	11 ...
10	100	110 ...
11	101	111 ...

until n

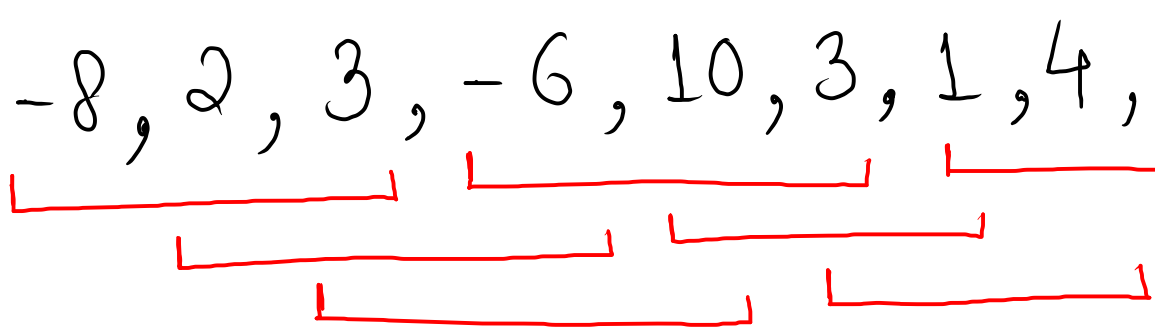
code

```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 str1 = rem + "0";
        que.add(str1);

        String str2 = rem + "1";
        que.add(str2);
    }
}
```

First Negative Integer 2

$K = 3$

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



ans = -8 -6 -6 -6 0 0 -2

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

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

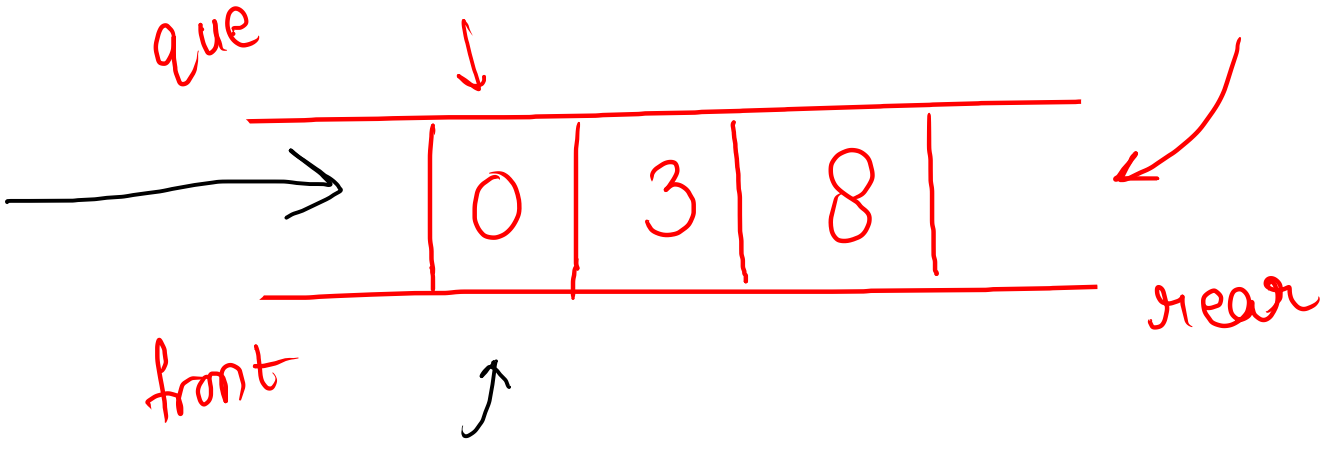
0 1 2 3 4 5 6 7 8

L

i

K = 3

step1) save index of first K -ve elements



```
if (que.size() > 0) {
        Syso(arr[que.peek()]);
} else {
        Syso(0);
}
```

print first -ve integer of window

```
while (que.peek() < (i - K + 1)) {
        que.poll();
}
if (arr[i] < 0) {
        que.add(i);
}
i++;
```

remove all elements which are out of window.

Code

```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;
    while (i < k) { // k times only (for first window)
        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  + " ");
        }

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

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

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