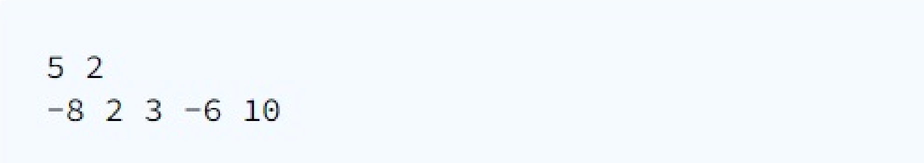


```
5 2
-8 2 3 -6 10
```

0 1 2 3 4

-8 2 3 -6 10

k = 2

N = 5

[-8, 2] → -8

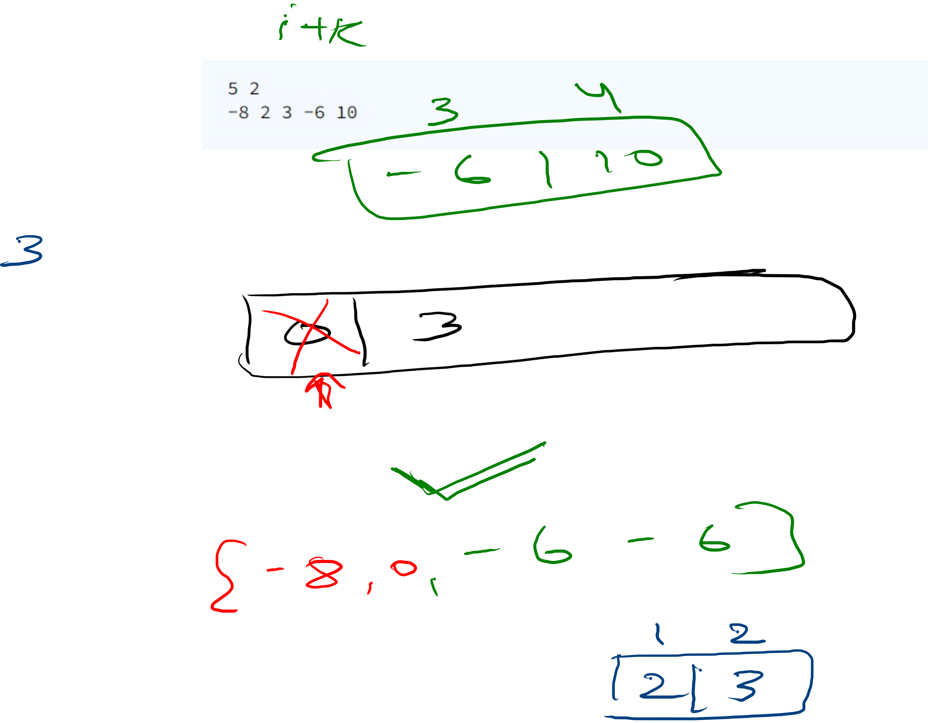
[2, 3] → 0

[3, -6] → -6

[-6, 10] → -6

| -8 | 0 | -6 | -

$\boxed{N - k + 1}$

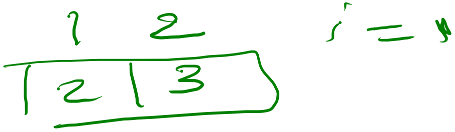


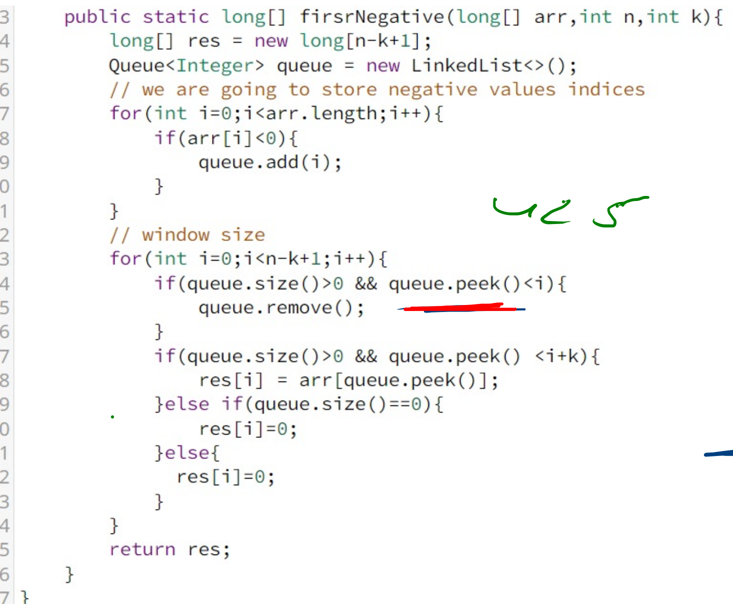
```
5 2
-8 2 3 -6 10
```

1. Create one queue

2. Store the negative elemenents Indexes.

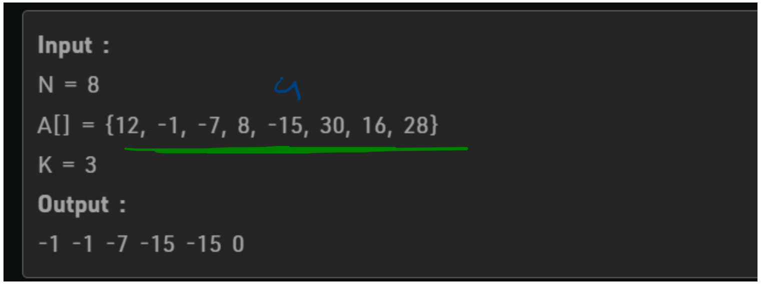
3. RUN one loop (0 to N-K+1)



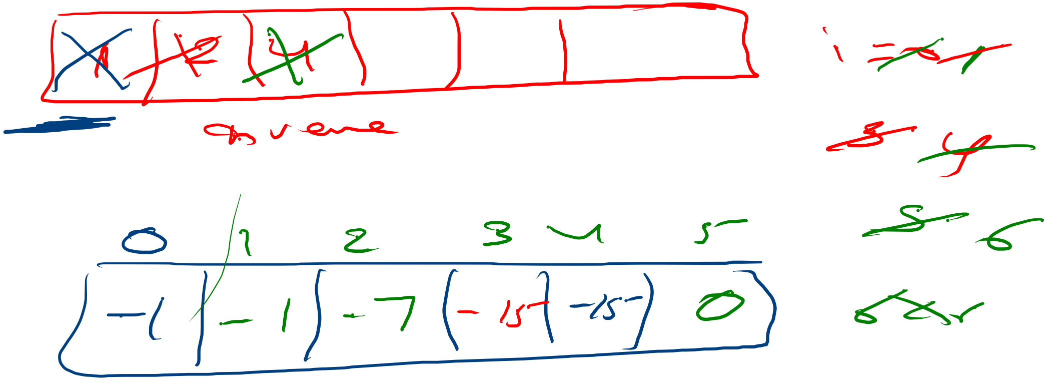


```
    public static long[] firsrNegative(long[] arr,int n,int k){
        long[] res = new long[n-k+1];
        Queue<Integer> queue = new LinkedList<>();
        // we are going to store negative values indices
        for(int i=0;i<arr.length;i++){
            if(arr[i]<0){
                queue.add(i);
            }
        }
        // window size
        for(int i=0;i<n-k+1;i++){
            if(queue.size()>0 && queue.peek()<i){
                queue.remove();
            }
            if(queue.size()>0 && queue.peek() <i+k){
                res[i] = arr[queue.peek()];
            }else if(queue.size()==0){
                res[i]=0;
            }else{
              res[i]=0;
            }
        }
        return res;
    }
}
```

4 2 5

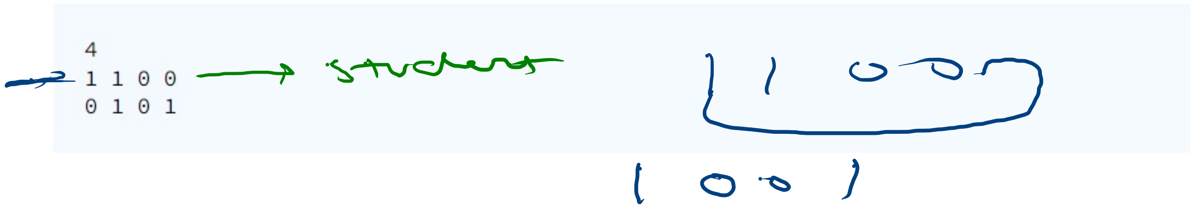


```
Input :
N = 8                  4
A[] = {12, -1, -7, 8, -15, 30, 16, 28}
K = 3
Output :
-1 -1 -7 -15 -15 0
```

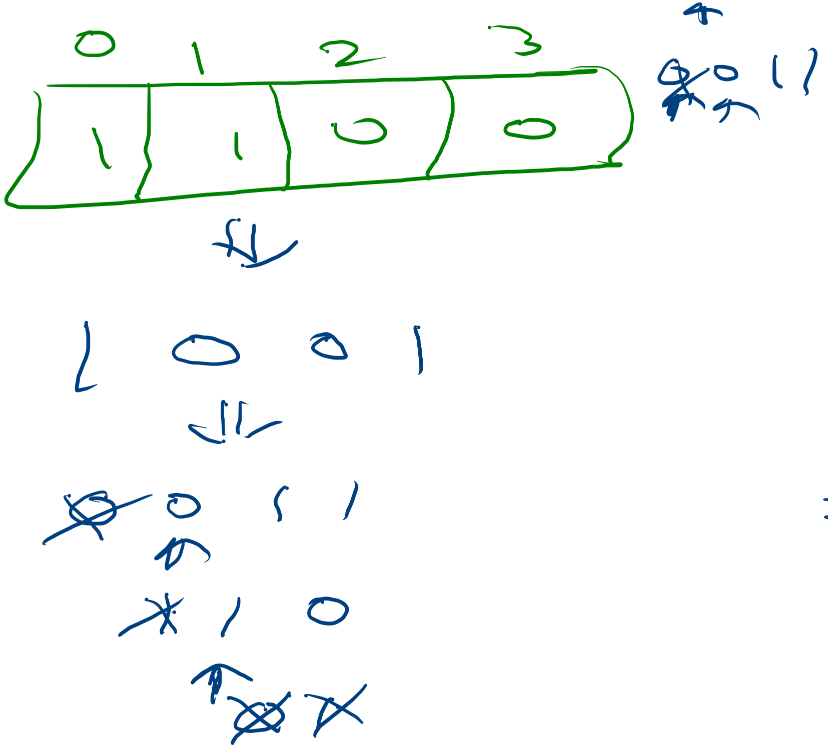


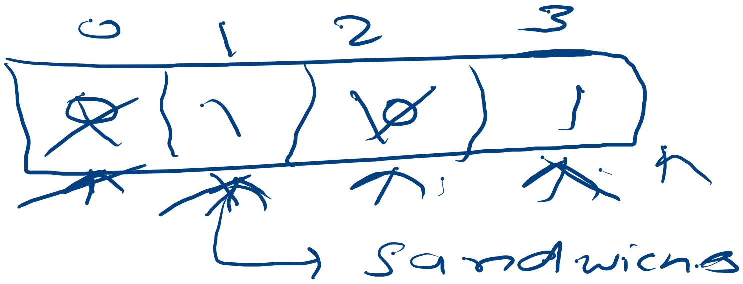
0 → circular

1 → square



```
4
1 1 0 0
0 1 0 1
```

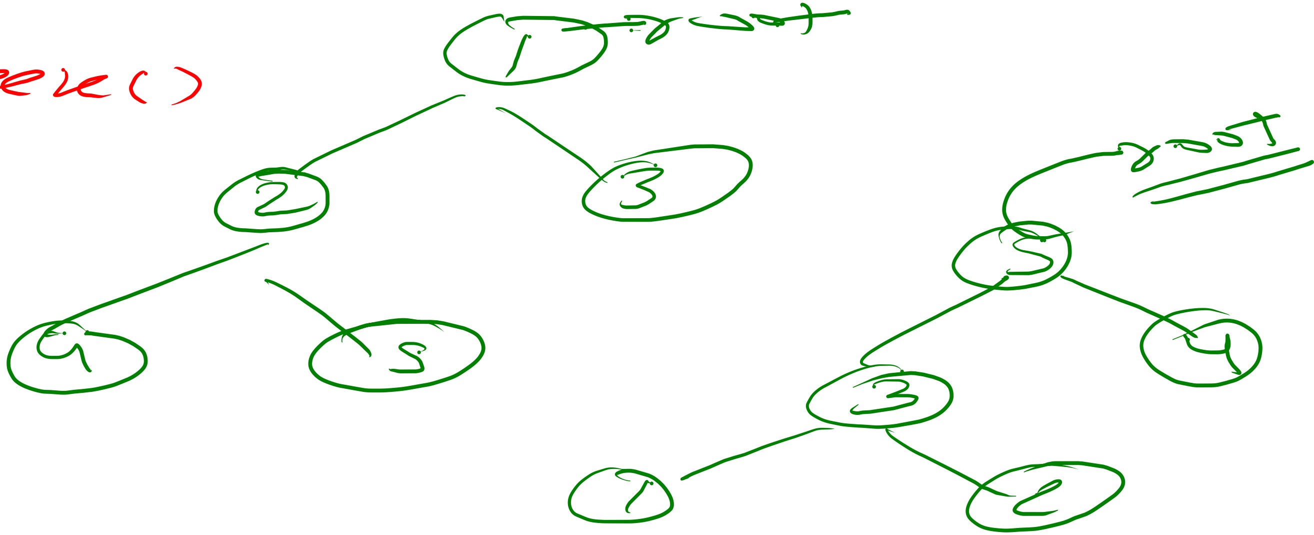


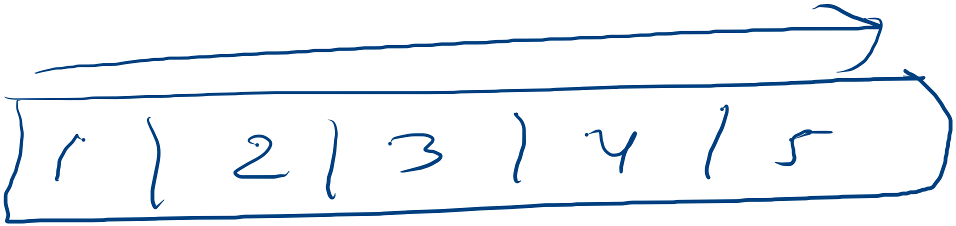


Priority Queue

- add
- remove
- poll
- peek()

[3, 5, 1, 2, 6]



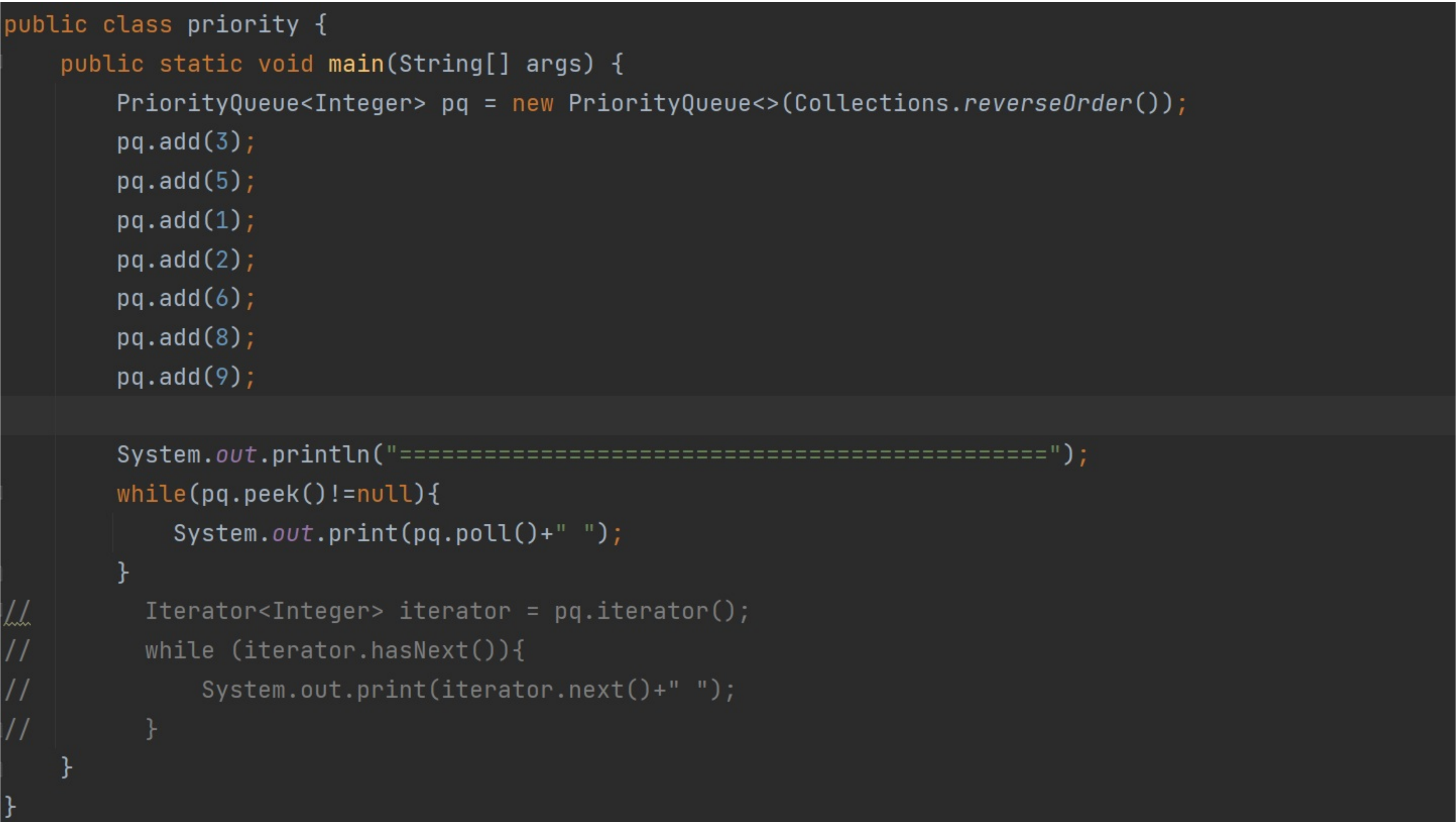


5

k = 2

[1, 2] [2, 3] [3, 4] [4, 5]

[0 0 0 0]



```java
public class priority {
    public static void main(String[] args) {
        PriorityQueue<Integer> pq = new PriorityQueue<>(Collections.reverseOrder());
        pq.add(3);
        pq.add(5);
        pq.add(1);
        pq.add(2);
        pq.add(6);
        pq.add(8);
        pq.add(9);

        System.out.println("==============================================");
        while(pq.peek()!=null){
            System.out.print(pq.poll()+" ");
        }
//        Iterator<Integer> iterator = pq.iterator();
//        while (iterator.hasNext()){
//            System.out.print(iterator.next()+" ");
//        }
    }
}
```