SLIDING WINDOW MAXIMUM

We are given an array of integers and an integer K which represents window size.

Our job is to return the array containing maximum elements from all possible windows of given size.

Brute force approach is to iterate through entire array, create a window for each possible element and find maximum everytime.

Time Complexity is $O(N-K) \times O(K)$
$= O(NK - K^2)$

Optimal Solution involves keeping track of K elements and using a monotonic stack (two-way)
The stack will store elements in decreasing order so that even when window is moved, order is maintained.

At any point, my dQ is not allowed to have more than K elements.

Pseudocode :

```
slidingWindowMaximum (arr, N, K) {
    ans[N-K];
    dq<int> d;
```

```
for(int i=0 → N-1) {
    if(!d.empty() && d.front() <= i-K) {
        d.pop_front();
    }
    while(!d.empty() && arr[d.back()]
        d.pop_back();                <= arr[i]){
    }
    d.push_back(arr[i]);
    if(i >= K-1) {
        ans[i-K-1] = arr[dq.front];
    }
}
return ans;
}
```