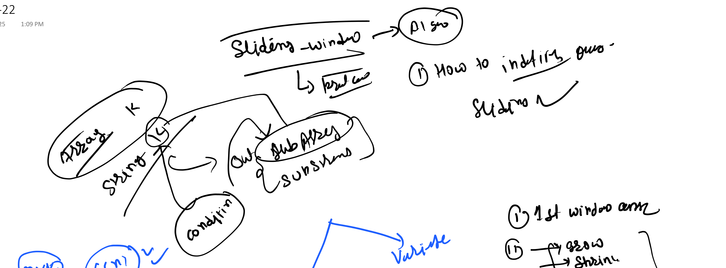
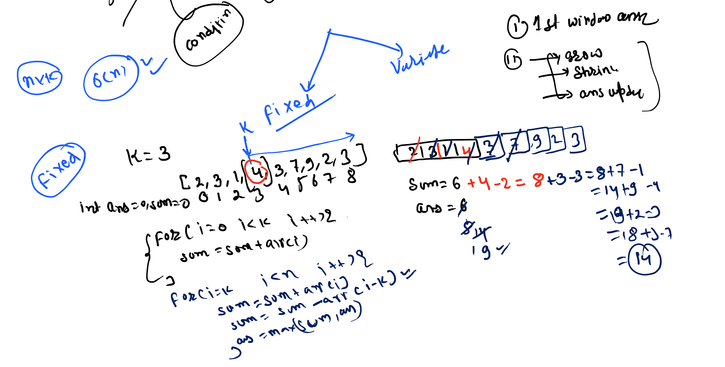
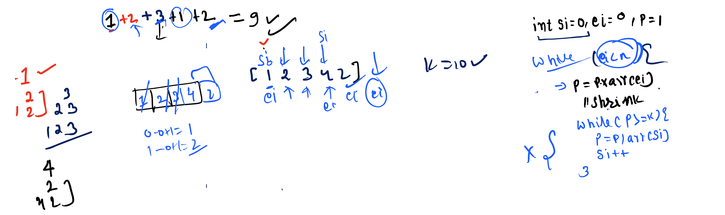
Lec-22

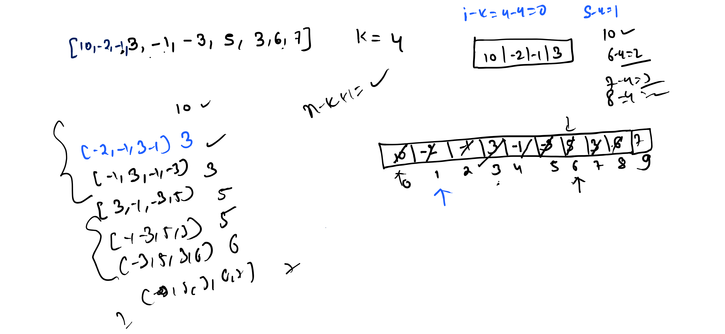
26/02/25 1:09 PM

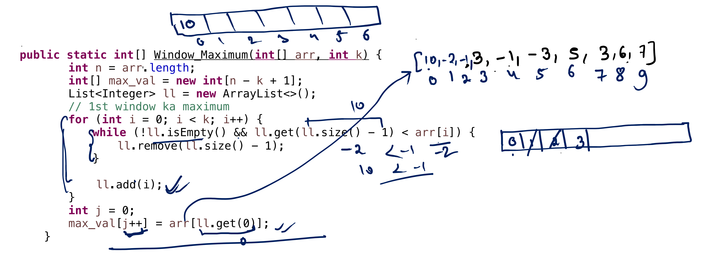




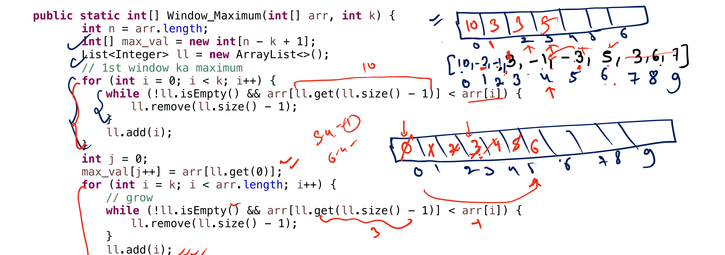



```
public static int Product_Less_Than_K(int[] arr, int k) {
        int si = 0, ei = 0, p = 1, ans = 0;
        while (ei < arr.length) {
            // grow
            p = p * arr[ei];

            // shrink
            while (p >= k) {
                p = p / arr[si];
                si++;
            }

            // ans update
            ans += (ei - si + 1);

            ei++;
        }
        return ans;
    }
```






```
public static int[] Window_Maximum(int[] arr, int k) {
        int n = arr.length;
        int[] max_val = new int[n - k + 1];
        List<Integer> ll = new ArrayList<>();
        // 1st window ka maximum
        for (int i = 0; i < k; i++) {
            while (!ll.isEmpty() && ll.get(ll.size() - 1) < arr[i]) {
                ll.remove(ll.size() - 1);
            }

            ll.add(i);
        }
        int j = 0;
        max_val[j++] = arr[ll.get(0)];
    }
```




```
public static int[] Window_Maximum(int[] arr, int k) {
        int n = arr.length;
        int[] max_val = new int[n - k + 1];
        List<Integer> ll = new ArrayList<>();
        // 1st window ka maximum
        for (int i = 0; i < k; i++) {
            while (!ll.isEmpty() && arr[ll.get(ll.size() - 1)] < arr[i]) {
                ll.remove(ll.size() - 1);
            }

            ll.add(i);
        }
        int j = 0;
        max_val[j++] = arr[ll.get(0)];
        for (int i = k; i < arr.length; i++) {
            // grow
            while (!ll.isEmpty() && arr[ll.get(ll.size() - 1)] < arr[i]) {
                ll.remove(ll.size() - 1);
            }
            ll.add(i);
            // shrink
            if (ll.get(0) == i - k) {
                ll.remove(0);
            }

            // ans upadte
            max_val[j++] = arr[ll.get(0)];

        }
        return max_val;

    }
```