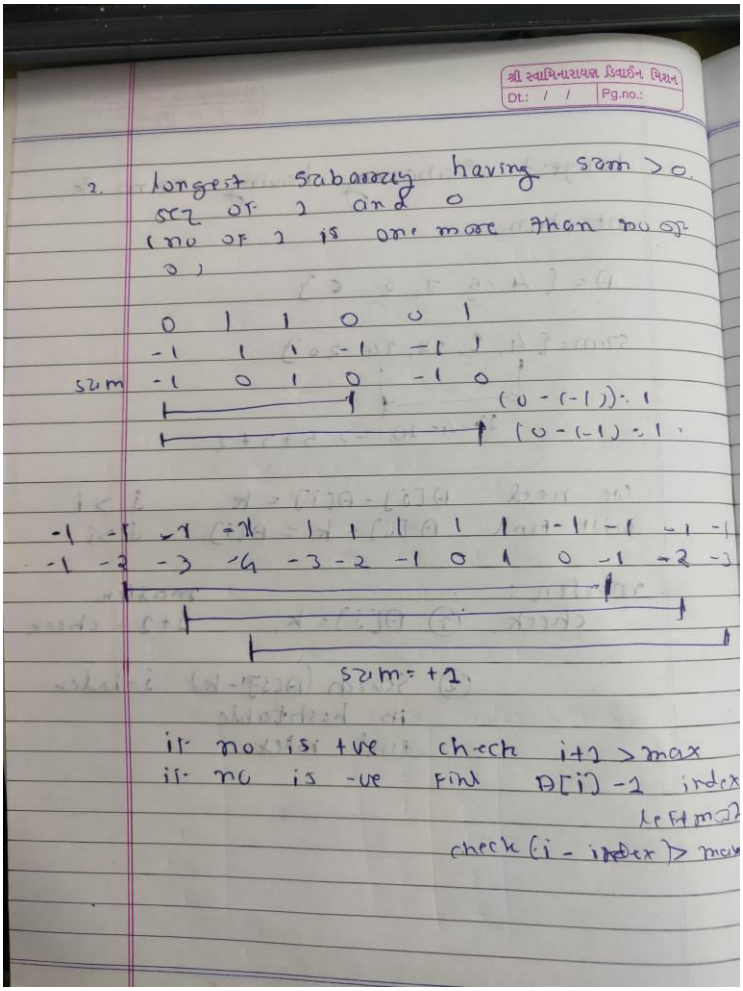
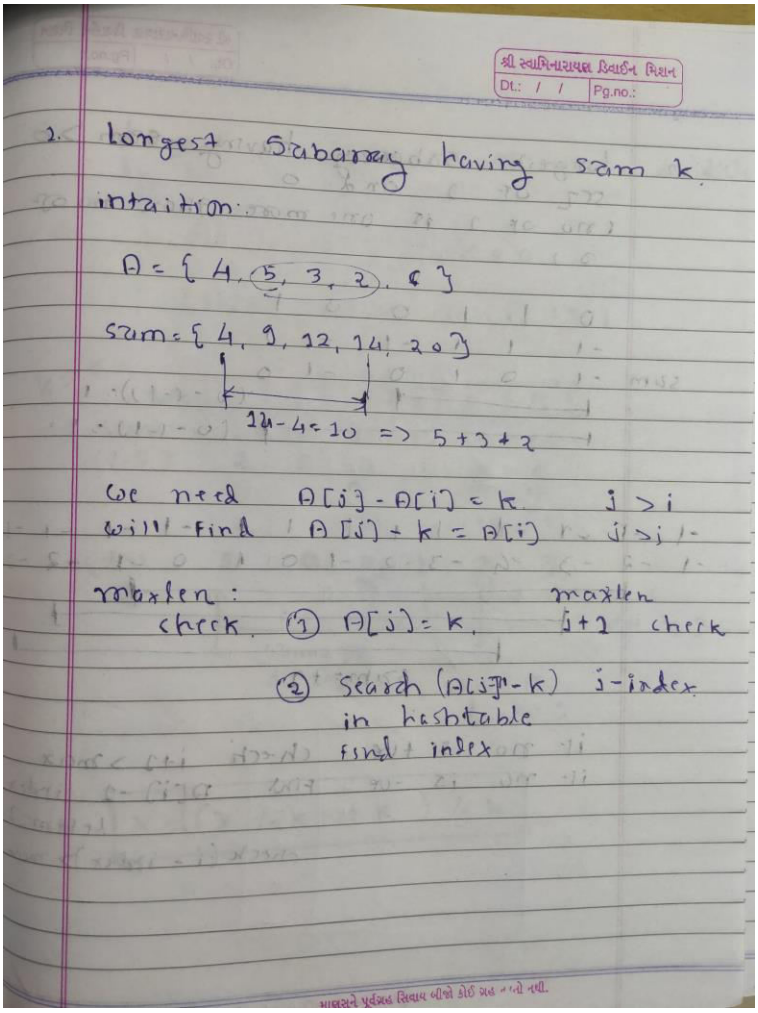
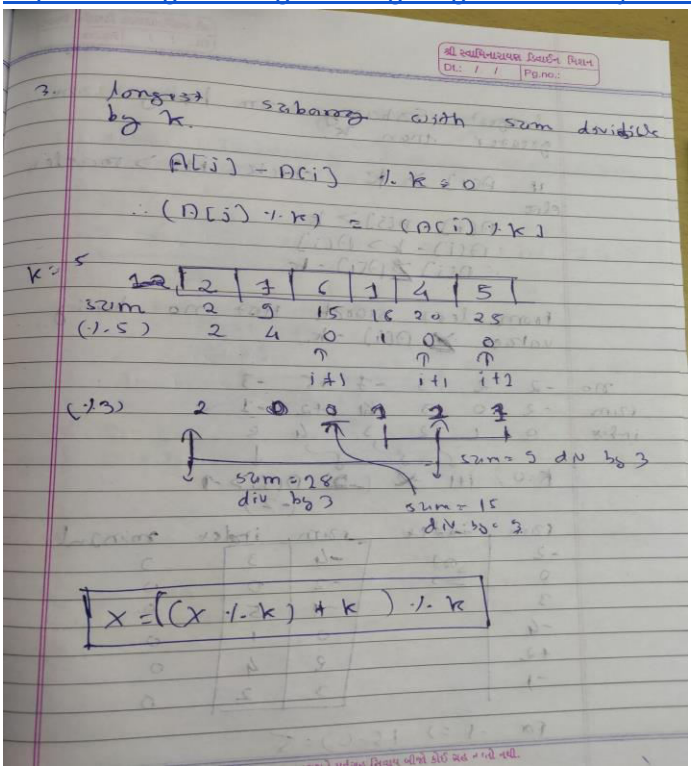
1. Longest subarray having sum > 0 in sequence of 0 and 1
   https://leetcode.com/problems/longest-well-performing-interval/submissions/



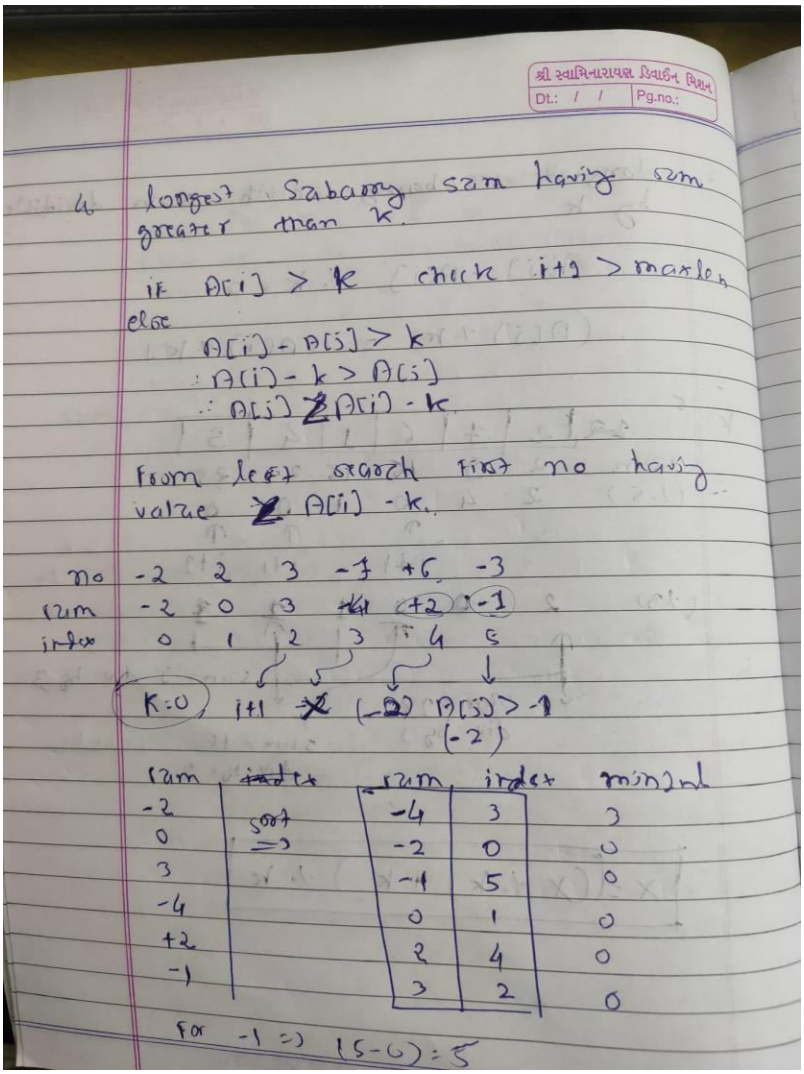
2. Longest Subarray having sum K
   https://www.interviewbit.com/problems/longest-subarray-sum-b/



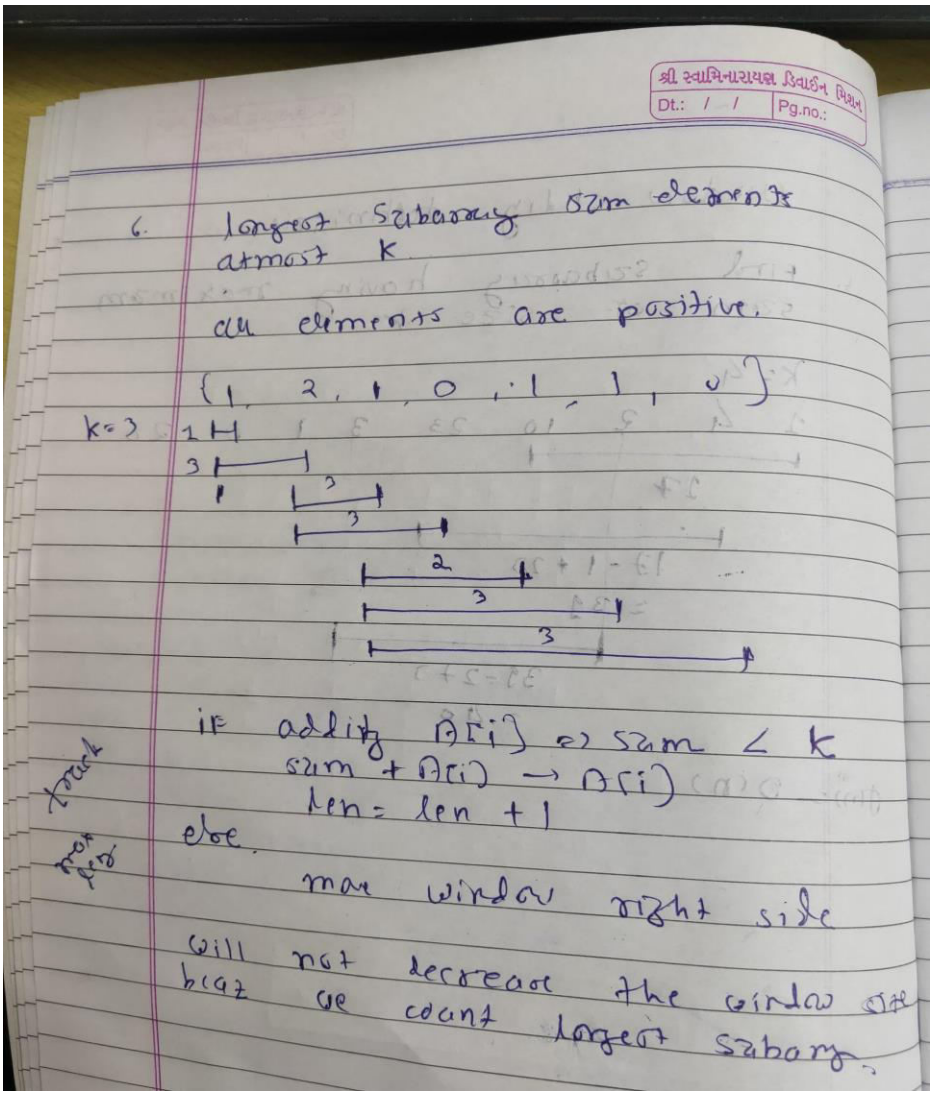
3. Longest Subarray sum divides k
   https://www.geeksforgeeks.org/longest-subarray-sum-divisible-k/



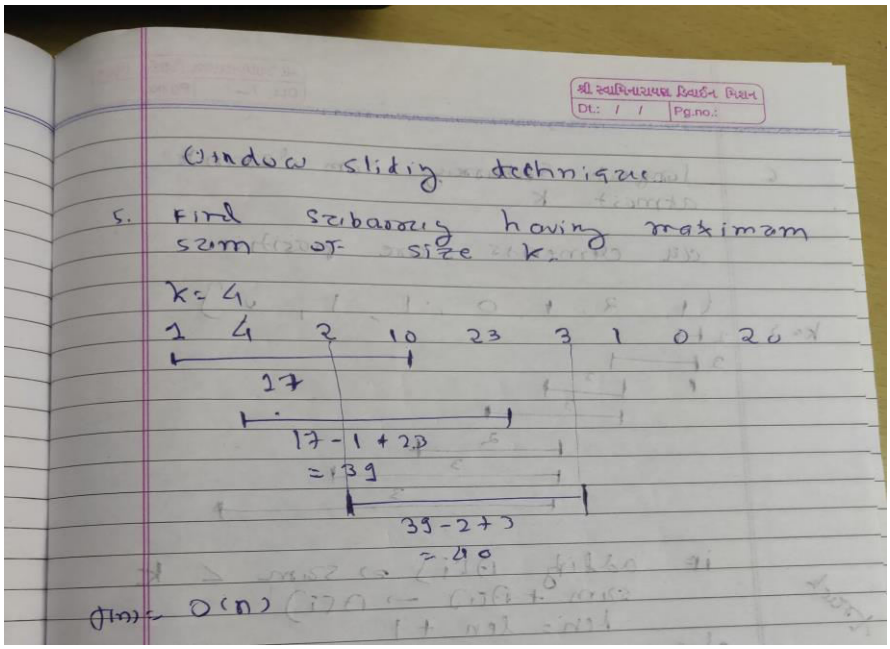

4. Longest Subarray sum having sum greater than k
   https://www.geeksforgeeks.org/longest-subarray-sum-divisible-k/




5. Sliding Window
    a. Longest Subarray sum at most k
    https://www.geeksforgeeks.org/longest-subarray-sum-elements-atmost-k/




b. Find subarray of size k having maximum sum
   i. https://www.geeksforgeeks.org/window-sliding-technique/




6. Longest Substring without repeating char
    a. https://leetcode.com/problems/longest-substring-without-repeating-characters/submissions/
    b. Use Sliding Window
        i. abcdec
        ii. abcdec
        iii. abcdecb

    Maintain a unordered map which maintains char to index mapping
    Starting index j and iterate through i
    When char is repeated make j = max(first occur index + 1, j)

7. Longest Subarray not having more than k distinct elements
    a. https://www.geeksforgeeks.org/longest-subarray-not-k-distinct-elements/
        i. 1 2 3 4 2 3 5 3 5
        ii. 1 2 3
        iii. 2 3 4 2 3
        iv. 2 3 5 3
        v. 2 3 5 3 5
    b. Idea is use unordered map whose size should be always <= k. If more than k inc i and remove ith element from map. Keep track of max len.
    c. i = starting window index , j = ending window index.
8. Maximum in each window of size k
    a. https://leetcode.com/problems/sliding-window-maximum/submissions/
    b. https://www.geeksforgeeks.org/sliding-window-maximum-maximum-of-all-subarrays-of-size-k-using-stack-in-on-time/

9. Maximum window substring
    a. https://leetcode.com/problems/minimum-window-substring/submissions/
    b. Use two pointers i = increment in map, j = decrement in map

        Initially size = t.length()

        If i increment from 0 then size++

        If j decrement to 0 then size--

        Check for min len while size = 0

    When size is 0 try to reduce window size from front

10. Maximum product subarray
    a. https://leetcode.com/problems/maximum-product-subarray/submissions/
    b. int a = maxp;

        int b = minp;

        maxp = max(nums[i], max(a*nums[i], b*nums[i]));

        minp = min(nums[i], min(a*nums[i], b*nums[i]));

        ans = max(ans, maxp);

11. https://leetcode.com/problems/longest-repeating-character-replacement/