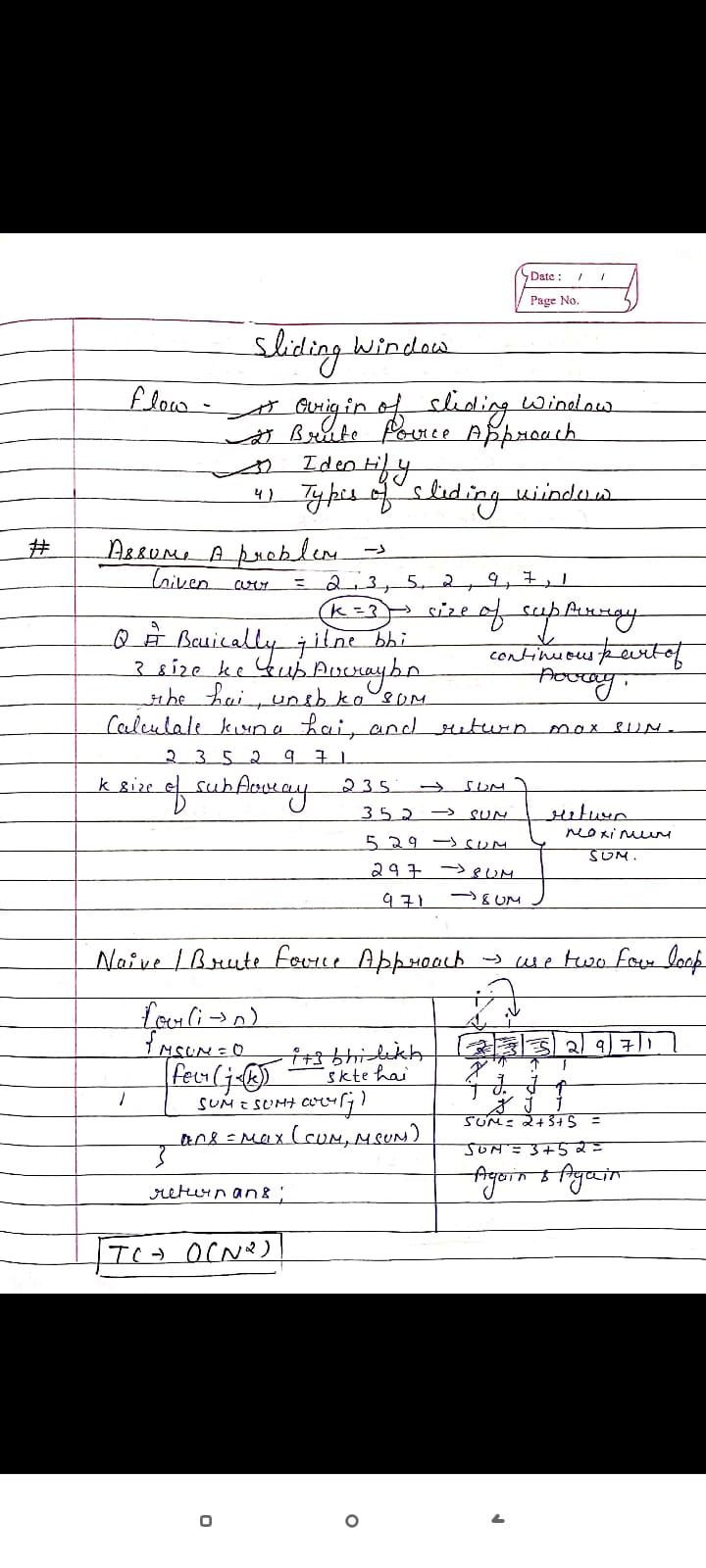
**# Code**

#include <bits/stdc++.h>

**using** **namespace** std;

// Returns maximum sum in a subarray of size k.

**int** maxSum(**int** arr[], **int** n, **int** k)

{

    // Initialize result

**int** max\_sum = INT\_MIN;

    // Consider all blocks starting with i., Hume Hr Jagh 3 element ka Pair le kr chalna pdega

**for** (**int** i = 0; i < n - k + 1; i++)

// n-k+1 because last me k size tk ka index bcha chaiye jbhi to hum sum kr paenge

{

**int** current\_sum = 0;

**for** (**int** j = 0; j < k; j++)

            current\_sum = current\_sum + arr[i + j];

        // Update result if required.

        max\_sum = max(current\_sum, max\_sum);

    }

**return** max\_sum;

}

// Driver code

**int** main()

{

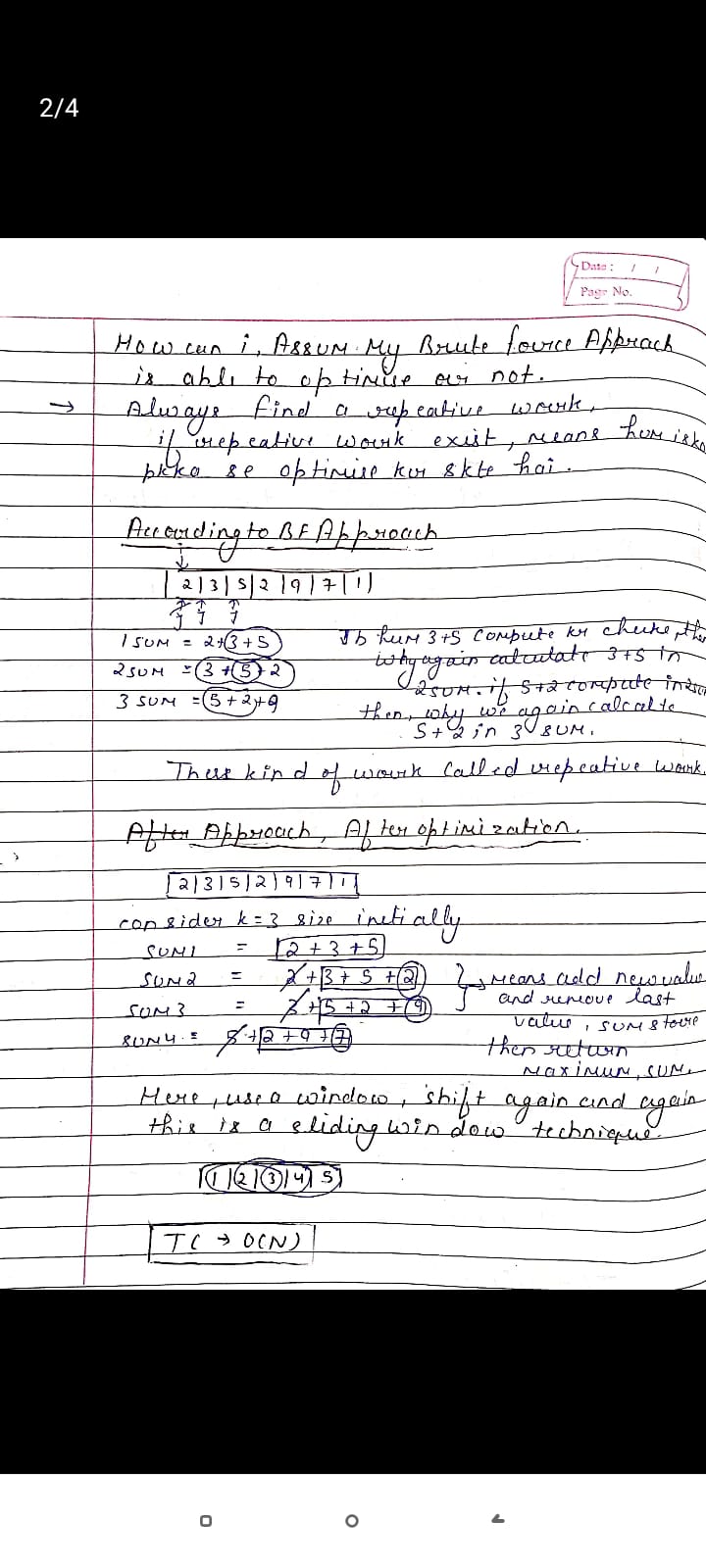
**int** arr[] = { 1, 4, 2, 10, 2, 3, 1, 0, 20 };

**int** k = 4;

**int** n = **sizeof**(arr) / **sizeof**(arr[0]);

    cout << maxSum(arr, n, k);

**return** 0;

}

**# Code**

#include <iostream>

**using** **namespace** std;

// Returns maximum sum in a subarray of size k.

**int** maxSum(**int** arr[], **int** n, **int** k)

{

    // n must be greater

**if** (n < k) {

        cout << "Invalid";

**return** -1;

    }

    // Compute sum of first window of size k

**int** max\_sum = 0;

**for** (**int** i = 0; i < k; i++)

        max\_sum += arr[i];

    // Compute sums of remaining windows by

    // removing first element of previous

    // window and adding last element of

    // current window.

**int** window\_sum = max\_sum;

**for** (**int** i = k; i < n; i++)

{

        window\_sum += arr[i] - arr[i - k];

        max\_sum = max(max\_sum, window\_sum);

    }

**return** max\_sum;

}

// Driver code

**int** main()

{

**int** arr[] = { 1, 4, 2, 10, 2, 3, 1, 0, 20 };

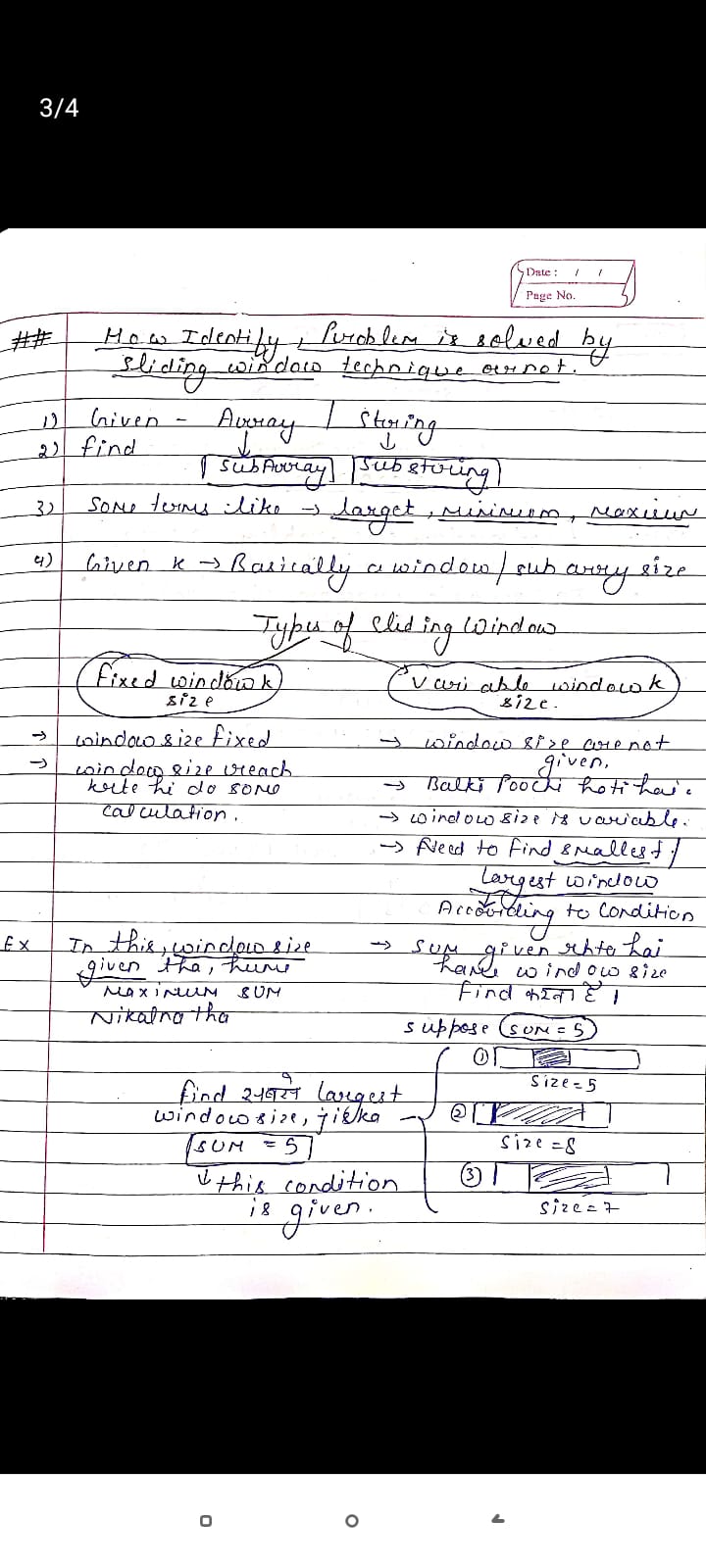
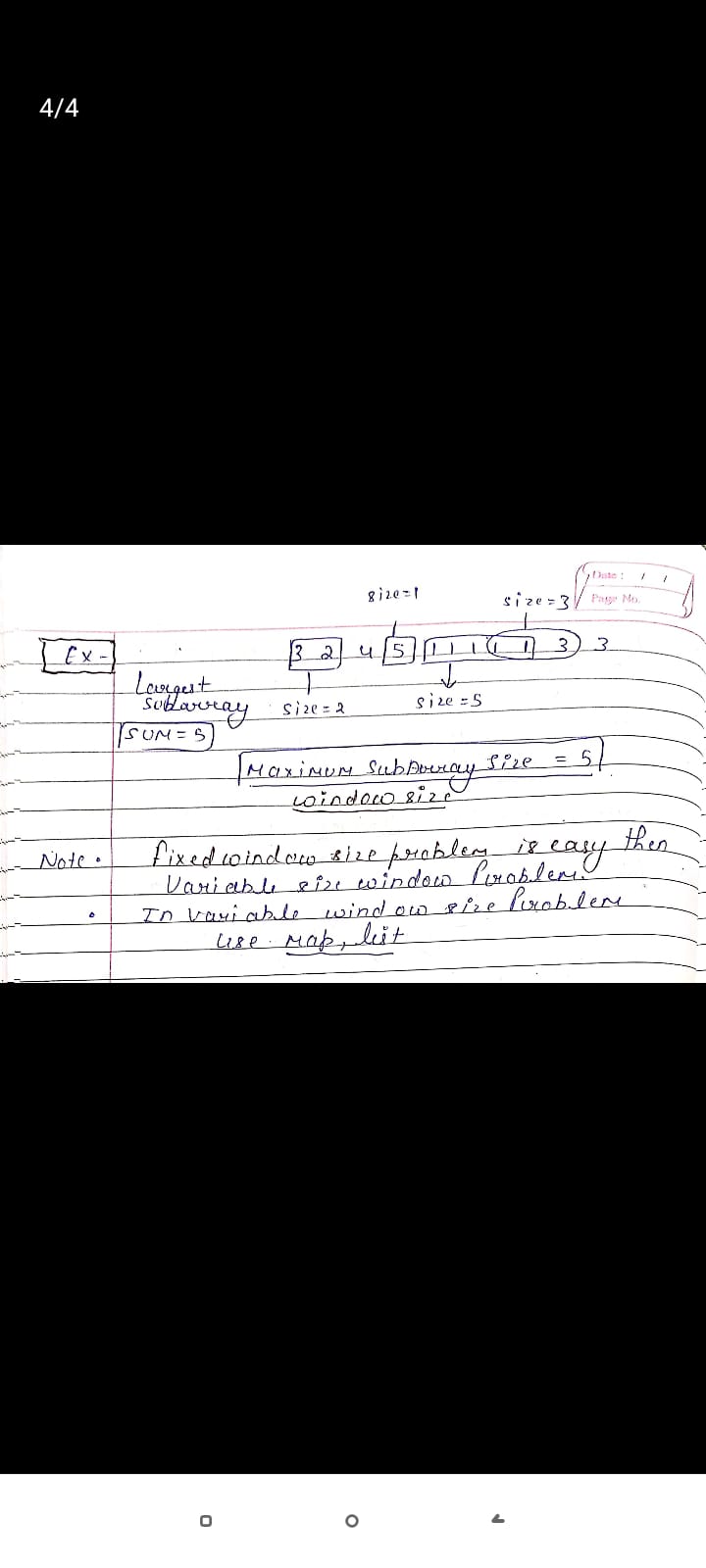
**int** k = 4;

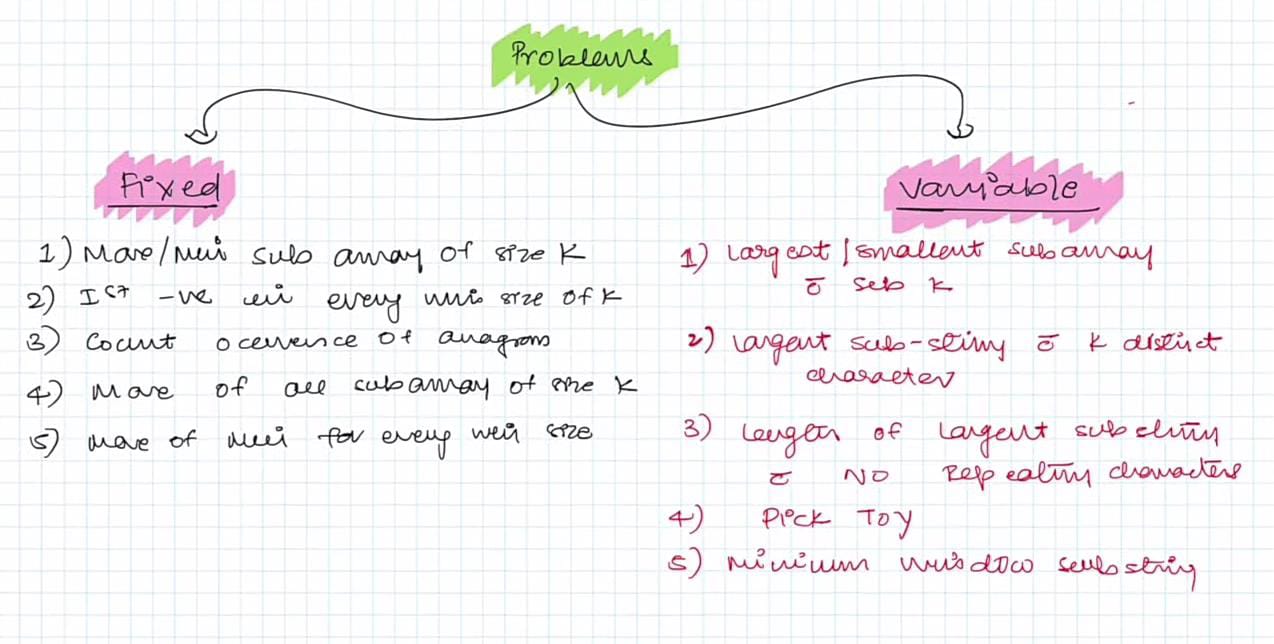
**int** n = **sizeof**(arr) / **sizeof**(arr[0]);

    cout << maxSum(arr, n, k);

**return** 0;

}

****

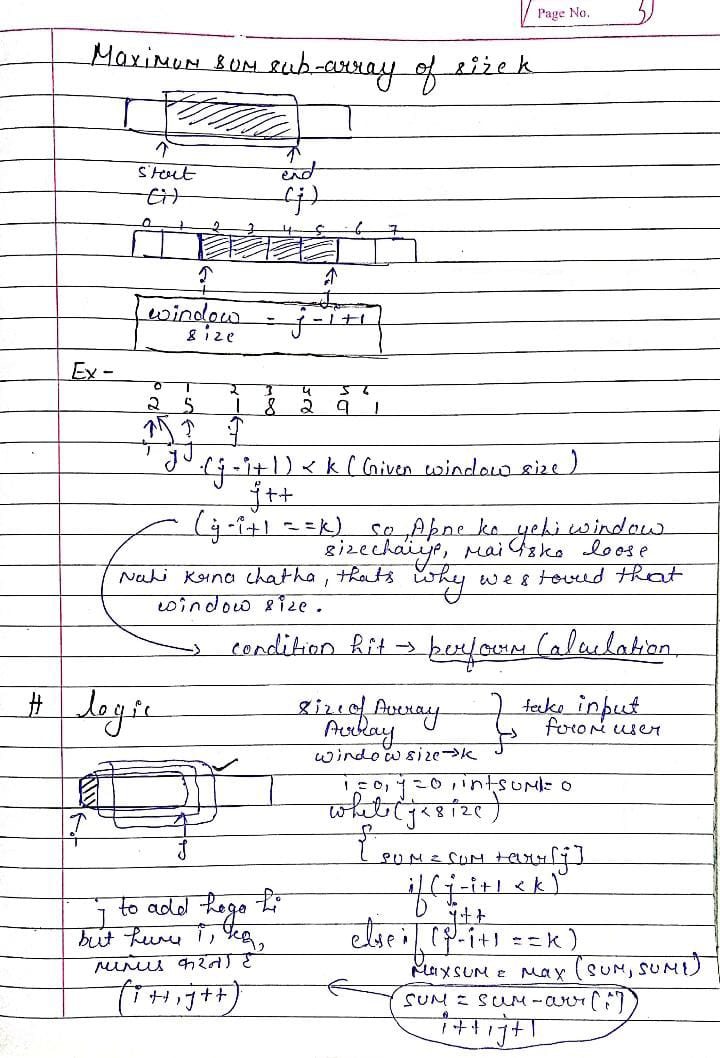
****

**1** [**Max Sum Subarray of size K**](https://practice.geeksforgeeks.org/problems/max-sum-subarray-of-size-k5313/1?utm_source=gfg&utm_medium=article&utm_campaign=bottom_sticky_on_article) **(Gfg)**

**1 Leetcode(** [**Maximum SubArray**](https://leetcode.com/problems/maximum-subarray/description/) **)**



**# 2ND Approach**

****