SLIDING WINDOW QUESTIONS

1.) LONGEST SUBSTRING WITHOUT REPEATING CHARACTERS

CODE

```cpp
class Solution {
public:
    int lengthOfLongestSubstring(string s) {
        int start=0;
        int end=0;
        int ans=0;
        unordered_map<char,int> mp;

        while(end<s.length()){
            while(mp[s[end]]>0){
                ans=max(ans,(end-start));
                mp[s[start]]--;
                start++;
            }
            mp[s[end]]++;
            end++;
        }
        ans=max(ans,(end-start));
        return ans;
    }
};
```

2.) MAX CONSECUTIVE ONES III

CODE

```
class Solution {
public:
    int longestOnes(vector<int>& nums, int k) {
        int i=0;
        int j=0;
        int ans=0;
        while(j<nums.size()){
            if(nums[j]==0)
            k--;
            while(k<0){
                if(nums[i]==0)
                k++;
                i++;
            }
            ans=max(ans,(j-i+1));
            j++;
        }
        return ans;
    }
};
```

3.) GET EQUAL SUBSTRING WITHIN BUDGETS

CODE

```
class Solution {
public:
    int longestOnes(vector<int>& nums, int k) {
        int i=0,j=0,ans=0;
        while(j<nums.size()){
            if(nums[j]==0)
            k--;
            while(k<0){
                if(nums[i]==0)
                k++;
                i++;
            }
            ans=max(ans,(j-i+1));
            j++;
        }
        return ans;
    }
};
```

4.) SUBARRAY PRODUCT LESS THAN K

CODE

```
class Solution {
public:
    int numSubarrayProductLessThanK(vector<int>& nums, int k) {
        int i=0;
        int j=0;
        int ans=0;
        int prod=1;

        // EDGE CASE
        if(k<=1)
        return 0;

        while(j<nums.size()){
            prod=prod*nums[j];
            while(prod>=k){
                prod=prod/nums[i];
                i++;
            }
            ans=ans+(j-i+1);
            j++;
        }
        return ans;
    }
};
```

5.) MAXIMUM ERASURE VALUE

CODE

```
class Solution {
public:
    int maximumUniqueSubarray(vector<int>& nums) {
        int i=0;
        int j=0;
        int sum=0;
        int ans=0;
        unordered_map<int,int> mp;

        while(j<nums.size()){
            while(mp[nums[j]]>0){
                mp[nums[i]]--;
                sum=sum-nums[i];
                i++;
            }
            sum=sum+nums[j];
            ans=max(ans,sum);
            mp[nums[j]]++;
            j++;
        }
        return ans;
    }
};
```

6.) LONGEST REPEATING CHARACTER REPLACMENT

CODE

```
class Solution {
public:
    int characterReplacement(string s, int k) {
        int i=0;
        int j=0;
        int ans=0;
        vector<int> count(26, 0);

        while(j<s.length()){
            count[s[j]-'A']++;
            while((j-i+1)-(*max_element(count.begin(),count.end()))>k){
                count[s[i]-'A']--;
                i++;
            }
            ans=max(ans,j-i+1);
            j++;
        }
        return ans;
    }
};
```

7.) MINIMUM SIZE SUBARRAY SUM

CODE

```
class Solution {
public:
    int minSubArrayLen(int target, vector<int>& nums) {
        int i=0;
        int j=0;
        int sum=0;
        int ans=INT_MAX;
        while(j<nums.size()){
            sum=sum+nums[j];
            while(sum>=target){
                ans=min(ans,(j-i+1));
                sum=sum-nums[i];
                i++;
            }
            j++;
        }
        if(ans==INT_MAX)
        return 0;
        return ans;
    }
};
```

8.) MINIMUM OPERATIONS TO REDUCE X TO ZERO

CODE

```cpp
class Solution {
public:
    int minOperations(vector<int>& nums, int x) {
        int sum=accumulate(nums.begin(),nums.end(),0);
        int req_sum=sum-x;
        if(req_sum==0)
        return nums.size();
        if(req_sum<0)
        return -1;

        int i=0;
        int j=0;
        int max_len=0;
        int my_sum=0;
        while(j<nums.size()){
            my_sum=my_sum+nums[j];
            while(my_sum>req_sum){
                my_sum=my_sum-nums[i];
                i++;
            }

            if(my_sum==req_sum)
            max_len=max(max_len, j-i+1);

            j++;
        }
        if(max_len==0)
        return -1;
        else
        return (nums.size()-max_len);
    }
};
```

9.) FIND ALL ANAGRAMS IN A STRING

CODE

```
class Solution {
public:
    vector<int> findAnagrams(string s, string p) {
        vector<int> hash(26, 0),temp(26, 0);

        for(int i=0;i<p.length();i++){
            hash[p[i]-'a']++;
        }
        int i=0;
        int j=0;
        int n=p.length();
        vector<int> ans;
        while(j<s.length()){
            if(temp==hash)
            ans.push_back(i);
            while(j-i+1>n){
                temp[s[i]-'a']--;
                i++;
            }
            temp[s[j]-'a']++;
            j++;
        }
        if(temp==hash)
        ans.push_back(i);
        return ans;
    }
};
```

10.) LONGEST SUBARRAY OF 1’S AFTER DELETING ONE ELEMENT

CODE

Same as max consecutive ones III

```
class Solution {
public:
    int longestSubarray(vector<int>& nums) {
        int i=0;
        int j=0;
        int ans=0;

        // small change
        int k=1;

        while(j<nums.size()){
            if(nums[j]==0)
            k--;
            while(k<0){
                if(nums[i]==0)
                k++;
                i++;
            }
            ans=max(ans,(j-i+1));
            j++;
        }
        return ans-1;
    }
};
```

11.) COUNT SUBARRAYS WITH SCORE LESS THAN K

CODE

```
class Solution {
public:
    long long countSubarrays(vector<int>& nums, long long k) {
        long long int ans=0;
        long long int i=0;
        long long int j=0;
        long long int sum=0;

        while(j<nums.size()){
            sum=sum+nums[j];

            while(sum*(j-i+1)>=k){
                sum=sum-nums[i];
                i++;
            }
            ans=ans+(j-i+1);
            j++;
        }
        return ans;
    }
};
```

12.) FRUITS INTO BASKETS

CODE

```
class Solution {
public:
    int totalFruit(vector<int>& items) {
        int i=0;
        int j=0;
        int count=0;
        unordered_map<int,int> mp;

        while(j<items.size()){
            mp[items[j]]++;
            while(mp.size()>2){
                mp[items[i]]--;

               // map m element jiski value zero h usko remove kr do
                if(mp[items[i]]==0)
                mp.erase(items[i]);

                i++;
            }
            count=max(count, j-i+1);
            j++;
        }
        return count;
    }
};
```

13.) MINIMUM CONSECUTIVE CARDS TO PICK UP

CODE

```
class Solution {
public:
    int minimumCardPickup(vector<int>& cards) {
        int i=0;
        int j=0;
        int ans=INT_MAX;
        unordered_map<int,int> mp;

        while(j<cards.size()){
            mp[cards[j]]++;

            while(mp[cards[j]]>1){
                ans=min(ans, j-i+1);
                mp[cards[i]]--;
                i++;
            }
            j++;
        }

        if(ans==INT_MAX)
        return -1;
        else
        return ans;
    }
};
```

14.) FREQUENCY OF MOST FREQUENT ELEMENT

CODE

```cpp
class Solution {
public:
    using ll= long long int;
    int maxFrequency(vector<int>& nums, int k) {
        sort(nums.begin(), nums.end());
        ll i=0;
        ll j=0;
        ll sum=0;
        ll ans=0;

        while(j<nums.size()){
            sum=sum+nums[j];

            if((j-i+1)*nums[j]-sum>k){
                sum=sum-nums[i];
                i++;
            }
            ans=max(ans, j-i+1);
            j++;
        }
        return ans;
    }
};
```

15.) NUMBER OF ZERO FILLED SUBARRAYS

CODE

```
class Solution {
public:
    long long zeroFilledSubarray(vector<int>& nums) {
        using ll=long long int;
        ll i=0;
        ll j=0;
        ll ans=0;

        while(j<nums.size()){
            i=j;
            while(j<nums.size() && nums[j]==0){
                ans=ans+(j-i+1);
                j++;
            }
            j++;
        }
        return ans;
    }
};
```

16.) NUMBER OF SMOOTH DESCENT PERIODS OF A STOCK

CODE

```
class Solution {
public:
    long long getDescentPeriods(vector<int>& prices) {
        using ll=long long int;
        ll i=0;
        ll j=1;
        ll ans=0;

        while(j<prices.size()){
            while(j<prices.size() && prices[j]-prices[j-1]==-1){
                ans=ans+(j-i);
                j++;
            }
            i=j;
            j++;
        }
        return ans+prices.size();
    }
};
```

17.) COUNT THE NUMBER OF GOOD SUBARRAYS

CODE

```cpp
class Solution {
public:
    long long countGood(vector<int>& nums, int k) {
        using ll=long long int;
        ll i=0;
        ll j=0;
        ll count=0;
        ll ans=0;
        unordered_map<ll, ll> mp;

        while(j<nums.size()){
            count=count+mp[nums[j]];
            mp[nums[j]]++;

            while(i<j && count>=k){
                ans=ans+(nums.size()-j);
                mp[nums[i]]--;
                count=count-mp[nums[i]];
                i++;
            }
            j++;
        }
        return ans;
    }
};
```

18.) LONGEST NICE SUBARRAYS

CODE

```
class Solution {
public:
    int longestNiceSubarray(vector<int>& nums) {
        int i=0;
        int j=0;
        int ans=0;
        int result=0;

        while(j<nums.size()){
            while((ans & nums[j])>0){
                ans=ans^nums[i];
                i++;
            }
            ans=ans|nums[j];
            result=max(result, j-i+1);
            j++;
        }
        return result;
    }
};
```

19.) MAXIMISE THE CONFUSION OF AN EXAM

CODE

```cpp
class Solution {
public:
    int maxConsecutiveAnswers(string s, int k) {
        int i=0;
        int j=0;
        int countT=0;
        int countF=0;
        int ans=0;

        while(j<s.size()){
            if(s[j]=='T')
            countT++;
            if(s[j]=='F')
            countF++;

            while(min(countT, countF)>k){
                if(s[i]=='T')
                countT--;
                if(s[i]=='F')
                countF--;
                i++;
            }
            ans=max(ans, j-i+1);
            j++;
        }
        return ans;
    }
};
```

20.) BINARY SUBARRAYS WITH SUM

CODE

```
class Solution {
public:
    int func(vector<int>& nums, int goal) {
        long long int i=0;
        long long int j=0;
        long long int ans=0;
        long long int sum=0;

        while(j<nums.size()){
            sum=sum+nums[j];
            while(i<=j && sum>goal){
                sum=sum-nums[i];
                i++;
            }
            ans=ans+(j-i+1);
            j++;
        }
        return ans;
    }

    int numSubarraysWithSum(vector<int>& nums, int goal) {
        return func(nums,goal)-func(nums,goal-1);
    }
};
```

21.) COUNT NUMBER OF NICE SUBARRAYS

CODE

```
class Solution {
public:
    int func(vector<int>& nums, int k) {
        int i=0;
        int j=0;
        int ans=0;

        while(j<nums.size()){
            if(nums[j]&1)
            k--;
            while(k<0){
                if(nums[i]&1)
                k++;
                i++;
            }
            ans=ans+(j-i+1);
            j++;
        }
        return ans;
    }
    int numberOfSubarrays(vector<int>& nums, int k) {
        return func(nums, k)-func(nums, k-1);
    }
};
```

22.) SUBARRAYS WITH K DIFFERENT INTEGERS

CODE

```
class Solution {
public:
    int func(vector<int> &nums, int k){
        int i=0;
        int j=0;
        int ans=0;
        unordered_map<int,int> mp;

        while(j<nums.size()){
            mp[nums[j]]++;

            while(mp.size()>k){
                mp[nums[i]]--;
                if(mp[nums[i]]==0)
                mp.erase(nums[i]);
                i++;
            }
            ans=ans+(j-i+1);
            j++;
        }
        return ans;
    }
    int subarraysWithKDistinct(vector<int>& nums, int k) {
        return func(nums, k)-func(nums, k-1);
    }
};
```

23.) MINIMUM SWAPS TO GROUP ALL 1'S TOGETHER II

CODE

```
class Solution {
public:
    int minSwaps(vector<int>& nums) {
        int count=0;
        for(int i=0;i<nums.size();i++){
            if(nums[i]==1)
            count++;
        }

        int w=count;
        int countZ=0;

        // initially zero handelled
        for(int i=0;i<w;i++){
            if(nums[i]==0)
            countZ++;
        }

        int mini=countZ;
        for(int i=w;i<w+nums.size();i++){
            if(nums[i%nums.size()]==0)
            countZ++;
            if(nums[i-w]==0)
            countZ--;

            mini=min(mini, countZ);
        }
        return mini;
    }
};
```

24.) MINIMUM WINDOW SUBSTRING

CODE

```
class Solution {
public:
    string minWindow(string s, string t) {
        int i=0;
        int j=0;
        int mini=INT_MAX;
        int start=0;
        int size=0;

        vector<int> hash(128);
        for(auto it:t){
            hash[it]++;
        }

        while(j<s.length()){

            hash[s[j]]--;
            if(hash[s[j]]>=0){
                size++;
            }

            while(size==t.size()){
                if(mini>(j-i+1)){
                    mini=j-i+1;
                    start=i;
                }
                hash[s[i]]++;
                if(hash[s[i]]>0)
                size--;
                i++;
            }

            j++;
        }

        if(mini==INT_MAX)
        return "";
        return s.substr(start, mini);
    }
};
```

25.) MAXIMUM NUMBER OF VOWELS IN A SUBSTRING OF GIVEN LENGTH

CODE

```
class Solution {
public:
    int maxVowels(string s, int k) {
        int i=0;
        int j=0;
        int count=0;
        int ans=0;

        while(j<s.length()){
            if(s[j]=='a' || s[j]=='e' || s[j]=='i' || s[j]=='o' || s[j]=='u'){
                count++;
            }
            if(j-i+1==k){
                ans=max(ans, count);
                if(s[i]=='a' || s[i]=='e' || s[i]=='i' ||
                   s[i]=='o' || s[i]=='u'){
                    count--;
                }
                i++;
            }
            j++;
        }
        return ans;
    }
};
```

26.) NUMBER OF SUBSTRINGS CONTAINING ALL THREE CHARACTERS

CODE

```
class Solution {
public:
    int numberOfSubstrings(string s) {
        int i=0;
        int j=0;
        int ans=0;
        unordered_map<int, int> mp;

        while(j<s.length()){
            mp[s[j]]++;

            while(mp['a'] && mp['b'] && mp['c']){
                ans=ans+(s.length()-j);
                mp[s[i]]--;
                i++;
            }

            j++;
        }
        return ans;
    }
};
```

27.) COUNT SUBARRAYS WHERE MAX ELEMENT APPEARS AT LEAST K TIMES

CODE

```
class Solution {
public:
    long long countSubarrays(vector<int>& nums, int k) {
        long long int maxi=*max_element(nums.begin(),nums.end());
        long long int i=0;
        long long int j=0;
        long long int ans=0;
        long long int count=0;
        while(j<nums.size()){
            if(nums[j]==maxi)
            count++;

            while(count>=k){
                if(nums[i]==maxi)
                count--;
                ans=ans+(nums.size()-j);
                i++;
            }
            j++;
        }
        return ans;
    }
};
```

28.) LENGTH OF LONGEST ALPHABETICAL CONTINUOUS SUBSTRING

CODE

```
class Solution {
public:
    int longestContinuousSubstring(string s) {
        int j=1;
        int maxi=1;
        int ans=1;
        while(j<s.length()){
            if(s[j]==s[j-1]+1){
                ans++;
                maxi=max(maxi, ans);
            }
            else
            ans=1;
            j++;
        }
        maxi=max(maxi, ans);
        return maxi;
    }
};
```

29.) SLIDING WINDOW MAXIMUM

CODE

```
class Solution {
public:
    vector<int> maxSlidingWindow(vector<int>& nums, int k) {
        vector<int> ans;
        deque<int> dq;
        for(int i=0;i<nums.size();i++){
            if(!dq.empty() && dq.front()==i-k){
                dq.pop_front();
            }
            while(!dq.empty() && nums[dq.back()]<=nums[i]){
                dq.pop_back();
            }
            dq.push_back(i);
            if(i>=k-1)
            ans.push_back(nums[dq.front()]);
        }
        return ans;
    }
};
```