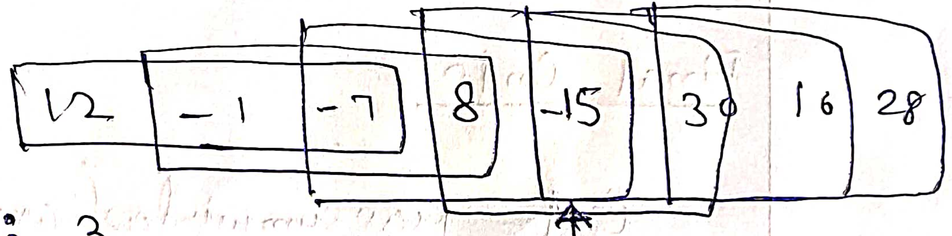
⇒ First Negative Number in every window of size K

1. PS - IP - OP
2. Expla~~tion~~nation < Brute force / Using prev Quest
3. Code

PS: IP: (arr[]:, arr size, window size)

arr[]:



K: 3

Output -1, -1, -7, -15, -15, 0

Given an array and a positive integer k, find the first negative integer for each and every window (contiguous subarray) of size K.

Example

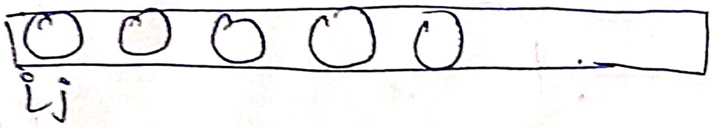
Input

2
5
-8 2 3 -6 10
2
8
12 -1 -7 8 -15 30 16 28
3

Output

-8, 0, -6, -6
-1, -1, -7, -15, -15, 0

Given size, arr, k→window size. (So this is Sliding window problem.



- we need to print size−k+1 number
- so we can apply sliding window.
- In interview 1st we will explain Brute force then gradually improve the solution

→ Brute force — for(i=0 — —) } repetitive work
→ Repetitive work for(j=0 — —) }
→ optimization < cond
→ which DS can be applied }
→ Go step by step

Identification → size of array, k→window size
→ Sub array ⇒ sliding window problem.

- we need to find the 1st negative.

```
If (j-i+1 < k)
    j++
else If (j-i+1 == k)
{
    1. Calculation for ans.
    2. Slide the window.
       i++ j++
}
```

① Find the Calculation ② Ans ← Calculation

③ Slide the window

```
        x               x
arr :  12   -1   -7    8   -15   30   16   28
        j    j    j    j

[ List   -1   -7   -15  - - - ]
```

```
While ( j < size )
{
    If ( arr[j] < 0 )
        l.push_back( ~~list~~ arr[j] )
    If ( j - i + 1 < k )
        j++;
    else If ( j - i + k == k )
    {
        1) Calculation
        If ( List.size() == 0 )   // Edge Case
            cout << 0 ;
        else {
            V.push_back( List.front() )     → take front in ans
        vector
```

Calculate ans

- If negative keep inside else j++;
- When window size hit and no negative Number return 0;

Calculation

Now we will remove 1st from the window and take next in order to maintain W size

```
        If ( arr[i] == l.front
             {l.pop.front();
             }
     }
  }   i++;   } // move actual window.
      j++;
```

```
      while ( j<size )
      {  Calculations
            If ( <K )
              j++
            If ( == K )
            {  1) ans -> Calculation
               2) Sliding the window
            }
      }
```

Sliding window Syntax.

Final Code

```
int i=0;
int j=0;

    vector <int> ans;
    vector <int> List;
     n= arr.size()

   While ( j< n)

  {  If (arr[j] < 0)  // If arr(i) is negative number
        List.pushback (arr[j] ) // take inside the List
   If ( j-i+1 < k)

          j++;
   else If ( j-i+1 == k )
   {
        If ( List.size () == 0) //Edge Case
             ans.push-back (0);
        else
           {ans.push-back ( List.front())
             If (arr[i] == List.front())
              {
                 List.pop.front();
              }
           }
        i++;
  }     j++;
}
```