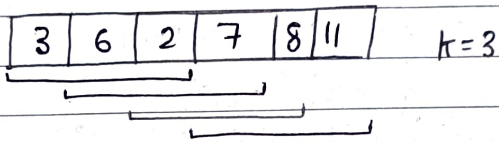
Date _____ Page _____

Day - 144

Queue - 3

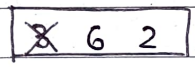
* **Print all no. in every window of size k:**



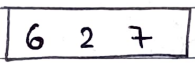
⇒ 3 6 2
6 2 7
2 7 8
7 8 11

⇒ We have to solve this question by using queue.

⇒ So,



→ 3 6 2



→ 6 2 7

and so on

⇒ So, we are popping out first element & pushing the next element.

⇒ First, we will push k-1 elements in the queue.

Date Page

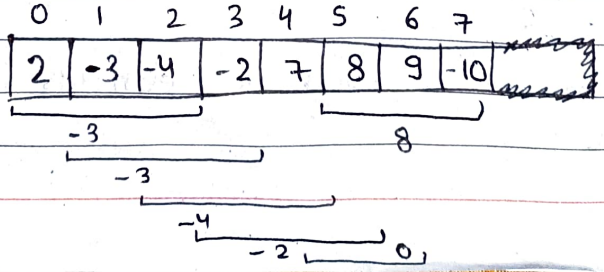
⇒ After that, we will push the elements & print then pop the first element & again push next element & so on. ⇒

Code

```
queue< int> q;
for( i=0; i<k-1; i++)
      q.push(arr[i]);
for( i= k-1; i<n; i++){
      q.push (arr[i]);
      display (q);
      q.pop();
}

void display (queue <int> q){
      while( !q.empty()){
            cout << q.front() << " ";
            q.pop();
      }
      cout << endl;
}
```

* First negative integer in every window of size k:



Date _____ Page _____

⇒ Now we will do that we have done in the previous question & we will get three elements.

⇒ And then we will print the negative element

Code

```
queue<int> q;
for(i=0; i<k-1; i++){
    q.push(arr[i]);
}
vector<int> ans;
for(i=k-1 ; i<n; i++){
    q.push(A[i]);
    ans.push_back(display(q));
    q.pop();
}
return ans;

int display(queue<int> q){
    while(!q.empty()){
        if(q.front()<0){
            return q.front();
        }
        q.pop();
    }
    return 0;
}
```

T.C. → O(nk)

Date _____ Page _____

⇉ Now, we have to solve it in O(n).

⇉ we will push only -ve no. into the queue.

⇉ Now, we will return the front element of the queue as first -ve no.

⇉ when, we solve this question, we will store indexing of the elements so it will be easier to check that element will in that window or not.

⇉ For checking, we will do that —
i - 3(k) =

⇉ If the index is equal to above value or less than we will pop it.

Code

```
queue <int>q;
for( i=0; i<k-1 ;i++){
        if( A[i] <0)
            q.push(i);
}
vector < int> ans;
for( i=k-1; i<n; i++){
    if( A[i]<0)
        q.push(i);
```

Date _____ Page _____

```
if(q.empty())
    ans.push_back(0);
else{
    if(q.front<=i-k)
        q.pop();
    if(q.empty())
        ans.push_back(0);
    else
        ans.push_back(A[q.front()]);
}
}
return ans;
```

* First non repeating char in a stream of chars:

A = "abab dc"

⇒ If the char is not repeated yet in that stream then add it to B.

⇒ If there is no such char then add '#'.

⇒ B = "aab#dd"

⇒ A = "abcacdbd"
B = "aaabbbd#

Date _____ Page _____

⇒ Now, we will solve it by optimized method.

⇒ We will take an ~~elem~~ array that stores the repeation of the elements.

⇒ We will use the queue to store the non repeated chars.

⇒ If queue becomes empty then add '#'.

⇒ Now, we will check every char that will come that it can be the answer. or not.

⇒ we will only push that char that ~~col~~ don't come yet.

Code

```
string B = " ";
vector<int> repeated(26,0);
queue<char> q;

for(int i= 0; i< A.size(); i++){
    if( repeated[A[i]-'a'] >= 1){
        repeated[A[i]-'a']++;
        while(!q.empty() && repeated[q.front()-'a']>1)
            q.pop();
        if(q.empty())
            B+='#';
        else{
            B+= q.front();
```

Date _____ Page _____

```
        else{
            repeated[A[i]-'a']++;
            q.push(A[i]);
            while(repeated[q.front()-'a']>1)
                q.pop();
            B += q.front();

        }
    }
    return B;
```