K sorted array in C++

```
#include <iostream>
#include <queue>
#include <vector>
using namespace std;

void sortKSortedArray(vector<int>& arr, int k) {
    priority_queue<int, vector<int>, greater<int>> pq; // Min heap

    // Push the first k+1 elements into the priority queue
    for (int i = 0; i <= k; ++i) {
        pq.push(arr[i]);
    }

    int index = 0;

    // Process the remaining elements
    for (int i = k + 1; i < arr.size(); ++i) {
        arr[index++] = pq.top(); // Get the smallest element from the heap
        pq.pop(); // Remove the smallest element from the heap
        pq.push(arr[i]); // Push the current element into the heap
    }

    // Extract all remaining elements from the heap
    while (!pq.empty()) {
        arr[index++] = pq.top();
        pq.pop();
    }

    // Print sorted array
    for (int i = 0; i < arr.size(); ++i) {
        cout << arr[i] << " ";
    }
    cout << endl;
}

int main() {
    // Hardcoded input array
    vector<int> arr = {7, 8, 9, 19, 18};
    int k = 3;

    // Sort the k-sorted array
    sortKSortedArray(arr, k);

    return 0;
}
```

Input:

```
arr = {7, 8, 9, 19, 18}
k = 3
```

We will walk through it step-by-step in a table format showing the **min heap**, **index**, and how the array is being modified.

🟦 Initial Step – Insert first `k+1 = 4` elements into min-heap:

Step	Action	Min Heap	arr[]	index
0	Insert first 4 elements (0–3)	[7, 8, 9, 19]	[7, 8, 9, 19, 18]	—

🔁 Main Loop (from i = k+1 to end):

Step	i	Action	Min Heap Before	Element Pushed	Popped → arr[index]	Min Heap After	arr[]	index
1	4	Push 18, Pop & insert 7	[7, 8, 9, 19]	18	7	[8, 18, 9, 19]	[7, 8, 9, 19, 18]	0
2	—	Pop & insert 8	[8, 18, 9, 19]	—	8	[9, 18, 19]	[7, 8, 9, 19, 18]	1
3	—	Pop & insert 9	[9, 18, 19]	—	9	[18, 19]	[7, 8, 9, 19, 18]	2
4	—	Pop & insert 18	[18, 19]	—	18	[19]	[7, 8, 9, 18, 18]	3
5	—	Pop & insert 19	[19]	—	19	[]	[7, 8, 9, 18, 19]	4

	✅ Final Output: 7 8 9 18 19
7 8 9 18 19	