Merge k sorted elements in C++

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

struct Pair {
    int li;   // List index
    int di;   // Data index (current index in the list)
    int val;  // Value at current index in the list

    Pair(int li, int di, int val) {
        this->li = li;
        this->di = di;
        this->val = val;
    }

    bool operator>(const Pair& other) const {
        return val > other.val;
    }
};

vector<int> mergeKSortedLists(vector<vector<int>>& lists) {
    vector<int> rv;

    // Min-heap priority queue
    priority_queue<Pair, vector<Pair>, greater<Pair>> pq;

    // Initialize the priority queue with the first element from each list
    for (int i = 0; i < lists.size(); ++i) {
        if (!lists[i].empty()) {
            pq.push(Pair(i, 0, lists[i][0]));
        }
    }

    while (!pq.empty()) {
        Pair p = pq.top();
        pq.pop();

        // Add the current value to result vector
        rv.push_back(p.val);

        // Move to the next element in the same list
        p.di++;
        if (p.di < lists[p.li].size()) {
            p.val = lists[p.li][p.di];
            pq.push(p);
        }
    }

    return rv;
}

int main() {
    vector<vector<int>> lists = {
        {10, 20, 30, 40, 50},
        {5, 7, 9, 11, 19, 55, 57},
        {1, 2, 3}
    };
```

Dry Run of mergeKSortedLists(lists)

Input:

```
lists = {
    {10, 20, 30, 40, 50},
    {5, 7, 9, 11, 19, 55, 57},
    {1, 2, 3}
};
```

Step 1: Initialize Min-Heap (priority_queue)

- Min-heap stores **(value, list index, data index)** for sorting.
- **Insert the first element of each list:**
  - (10, 0, 0) from list **0** ({10, 20, 30, 40, 50})
  - (5, 1, 0) from list **1** ({5, 7, 9, 11, 19, 55, 57})
  - (1, 2, 0) from list **2** ({1, 2, 3})

Step 2: Extract Minimum & Insert Next Element

Step	Extracted (Min)	Insert Next	Updated Heap
1	(1, 2, 0)	(2, 2, 1)	{(2,2,1), (5,1,0), (10,0,0)}
2	(2, 2, 1)	(3, 2, 2)	{(3,2,2), (5,1,0), (10,0,0)}
3	(3, 2, 2)	None (End)	{(5,1,0), (10,0,0)}
4	(5, 1, 0)	(7, 1, 1)	{(7,1,1), (10,0,0)}
5	(7, 1, 1)	(9, 1, 2)	{(9,1,2), (10,0,0)}
6	(9, 1, 2)	(11, 1, 3)	{(10,0,0), (11,1,3)}
7	(10, 0, 0)	(20, 0, 1)	{(11,1,3), (20,0,1)}
8	(11, 1, 3)	(19, 1, 4)	{(19,1,4), (20,0,1)}
9	(19, 1, 4)	(55, 1, 5)	{(20,0,1), (55,1,5)}
10	(20, 0, 1)	(30, 0, 2)	{(30,0,2), (55,1,5)}
11	(30, 0, 2)	(40, 0, 3)	{(40,0,3), (55,1,5)}
12	(40, 0, 3)	(50, 0, 4)	{(50,0,4), (55,1,5)}
13	(50, 0, 4)	None (End)	{(55,1,5)}
14	(55, 1, 5)	(57, 1, 6)	{(57,1,6)}
15	(57, 1, 6)	None (End)	{}

Final Merged List:

{1, 2, 3, 5, 7, 9, 10, 11, 19, 20, 30, 40, 50, 55, 57}

```
    vector<int> mlist = mergeKSortedLists(lists);

    for (int val : mlist) {
        cout << val << " ";
    }
    cout << endl;

    return 0;
}
```

Output:
1 2 3 5 7 9 10 11 19 20 30 40 50 55 57