C++ Assignment Solutions | Priority Queues 1 | Week 20

1. Kth largest element in a stream LEETCODE:-703

   Sol:

```
class KthLargest {
public:
    priority_queue<int,vector<int>,greater<int>>minHeap;
    int k;
    KthLargest(int k, vector<int>& nums) {
        this->k = k;
        int i = 0;
        for(i = 0;i < k && i < nums.size();i++){
            minHeap.push(nums[i]);
        }
        while(i < nums.size()){
            if(minHeap.top() < nums[i]){
                minHeap.pop();
                minHeap.push(nums[i]);
            }
            i++;
        }
    }
```

```
    int add(int val) {
        if(minHeap.size() < k){
            minHeap.push(val);
        }
        else if(minHeap.top() < val){
            minHeap.pop();
            minHeap.push(val);
        }
        return minHeap.top();
    }
};
```

2. K closest points to origin LEETCODE:-973

   Sol:

```
class Solution {
public:
    vector<vector<int>> kClosest(vector<vector<int>>& points, int k) {
        int n = points.size();
        priority_queue<pair<int, pair<int, int>>, vector<pair<int, pair<int,
int>>>, greater<pair<int, pair<int, int>>>> pq;
        for(auto it : points){
            int dist = it[0]*it[0] + it[1]*it[1];
            pq.push({dist, {it[0], it[1]}});
        }
        vector<vector<int>> ans;
        while(k--){
            int row = pq.top().second.first;
            int col = pq.top().second.second;
            pq.pop();
            ans.push_back({row, col});
        }
        return ans;
    }
};
```

3. Merge k sorted lists LEETCODE:-23

   Sol:

```
class Solution{
public:
    ListNode *mergeKLists(vector<ListNode *> &lists){
        priority_queue<int, vector<int>, greater<int>> min_heap;
        for (auto list : lists){
            while (list){
                min_heap.push(list->val);
                list = list->next;
            }
        }
        if (min_heap.empty()){
            return nullptr;
        }
        ListNode *res = new ListNode(min_heap.top()); min_heap.pop();
        ListNode *current = res;
        while (!min_heap.empty()){
            current->next = new ListNode(min_heap.top());
            current = current->next;
            min_heap.pop();
        }
        return res;
    }
};
```

Note:- Please try to invest time doing the assignments which are necessary to build a strong foundation. Do not directly Copy Paste using Google or ChatGPT. Please use your brain😀.