Assignment 1 priority_queue

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

Code:-

```
class KthLargest {
public:
int k;
 priority_queue<int,vector<int>,greater<int>>pq;
    KthLargest(int K, vector<int>& nums) {
        k=K;
        for(auto x:nums){
            if(pq.size()<k)pq.push(x);
            else{
            if(x>pq.top()){
                pq.pop();
            pq.push(x);
            }
            }
        }
    }

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

};
```

Q.2 K closest points to origin LEETCODE:-973

Code:-

```
class Solution {
public:
    vector<vector<int>> kClosest(vector<vector<int>>& points, int k) {
        vector<vector<int>>ans;
        priority_queue<pair<int,pair<int,int>>>pq;
        for(int i=0;i<points.size();i++){
            int dis=points[i][0]*points[i][0]+points[i][1]*points[i][1];
            if(pq.size()<k)pq.push({dis,{points[i][0],points[i][1]}});
            else{
            if(dis<pq.top().first){
                pq.pop();
                pq.push({dis,{points[i][0],points[i][1]}});
            }

            }
        }
            while(!pq.empty()){
                ans.push_back({pq.top().second.first,pq.top().second.second});
                pq.pop();

        }
        return ans;
    }
};
```

Q.3 Merge k sorted lists LEETCODE:-23

```
class Solution {
public:
    vector<vector<int>> kClosest(vector<vector<int>>& points, int k) {
        vector<vector<int>>ans;
        priority_queue<pair<int,pair<int,int>>>pq;
        for(int i=0;i<points.size();i++){
            int dis=points[i][0]*points[i][0]+points[i][1]*points[i][1];
            if(pq.size()<k)pq.push({dis,{points[i][0],points[i][1]}});
            else{
            if(dis<pq.top().first){
                pq.pop();
                pq.push({dis,{points[i][0],points[i][1]}});
            }

            }
        }
            while(!pq.empty()){
                ans.push_back({pq.top().second.first,pq.top().second.second});
                pq.pop();

        }
        return ans;
    }
};
```