H.K

Q Heap 1.

→ problem on Hackerrank involves implementing a priority queue (min-heap) with the following Operations :→

1. Add an element to the heap.
2. Delete an element from the heap (not necessarily the minimum).
3. Print the minimum element in the heap.

→ Approach

The key challenge is efficiently handling deletion of Arbitrary elements while maintaining the heap property. This can be done using :

- A min-heap for efficient extraction of the minimum element.
- A lazy deletion mechanism using a hash table (or map) to keep track of element that should be ignored in heap.

C++ Solution:→

Here is an efficient implementation using a min-heap and unordered-map for lazy deletions :→

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

int main() {
    int q; // Number of queries
    cin >> q;
    priority_queue <int, vector<int>, greater<int>> minHeap; // Min Heap
    unordered_map <int, int> deleteMap; // Tracks elements to delete.

    while (q--) {
    int type;
    cin >> type;

    if (type==1){
      // Insert an element
     int x;
     cin >> x;
     minHeap.push(x);
    } else if (type==2){
      // Mark an element for deletion
      int x;
      cin >> x;
      deleteMap[x]++;
    } else if (type==3) {  // get the minimum element, clean up lazy deletion
      while (!minHeap.empty() && deleteMap[minHeap.top()]>0) {
          deleteMap[minHeap.top()]--;
          minHeap.pop();
        }
      if (!minHeap.empty()){
          cout << minHeap.top() << endl;
    }
}
```

H·K

Example :→

i/p

queries 10

1	4
1	9
1	7
3	
2	4
3	
1	2
3	
2	2
3	

o/p

4
7
2
7

Explanation

1. Insert 4 : Heap = [4]
2. Insert 9 : Heap = [4,9]
3. Insert 7: Heap = [4,9,7]
4. Query 3: Print the min. element, which is 4
5. Delete 4 : Mark 4 for deletion. Heap is not adjusted yet ; lazy deletion used.
6. Query 3 : clean up the heap (remove 4). Now Heap = [7,9]. Print 7.
7. Insert 2 : Heap = [2,9,7]
8. Query 3 : Print the min element, which is 2.
9. Delete 2 : Mark 2 for deletion.
10. Query 3 : clean up the heap (remove 2). Now Heap = [7,9]. Print 7.