Experiment 1.3

Student Name: Md Tofique Alam **UID: 20BCS5508**
Branch: CSE **Section/Group: 20BCS_DM-901/A**
Semester: 6th **Date of Performance: 09/03/2023**
Subject Name: Competitive Coding-II **Subject Code: 20CSP-351**

Aim: Kth Largest Element in a Stream

Design a class to find the kth largest element in a stream. Note that it is the kth largest element in the sorted order, not the kth distinct element.

Implement KthLargest class:

KthLargest(int k, int[] nums) Initializes the object with the integer k and the stream of integers nums.

int add(int val) Appends the integer val to the stream and returns the element

representing the kth largest element in the stream.

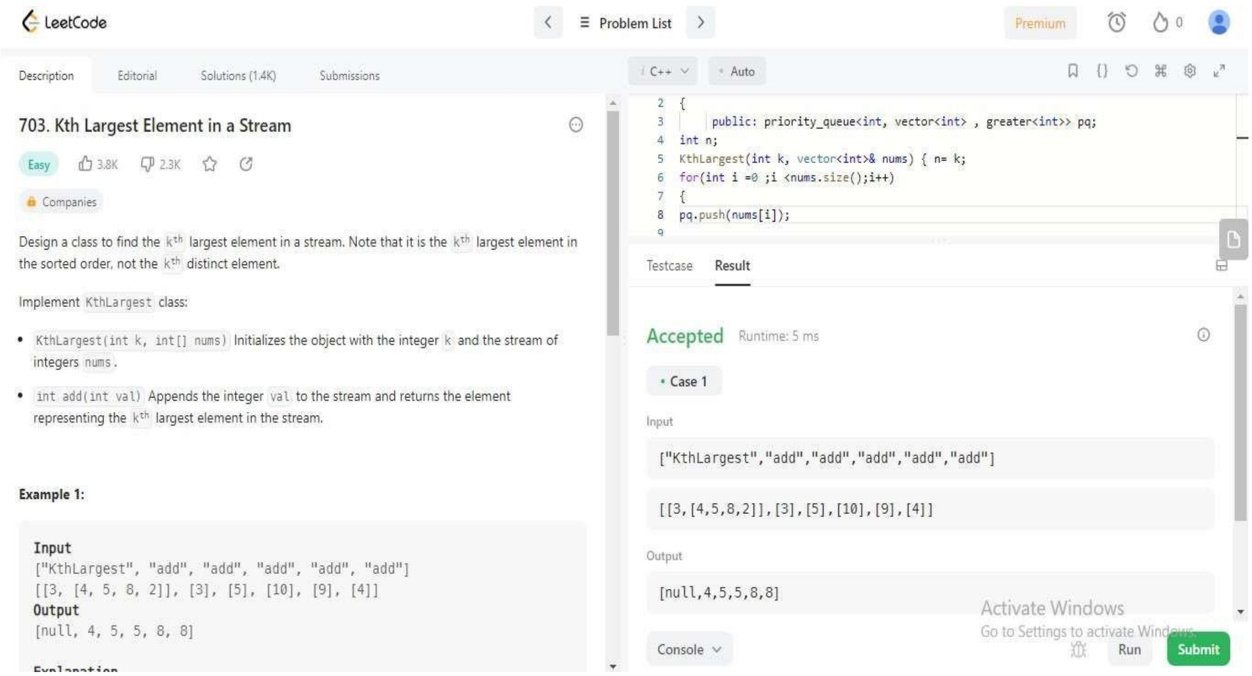
Program:

```
class KthLargest
{ public: priority_queue<int, vector<int> , greater<int>> pq;
int n;
   KthLargest(int k, vector<int>& nums) { n= k;
      for(int i =0 ;i <nums.size();i++)
      { pq.push(nums[i]);


          if(pq.size()>n)
pq.pop();
      }
```

```
    }          int add(int val)
  {          pq.push(val);
            if(pq.size()>n)
pq.pop(); return
         pq.top();
    }
};
```

Output:



703. Kth Largest Element in a Stream

Design a class to find the `k`th largest element in a stream. Note that it is the `k`th largest element in the sorted order, not the `k`th distinct element.

Implement `KthLargest` class:

- `KthLargest(int k, int[] nums)` Initializes the object with the integer `k` and the stream of integers `nums`.
- `int add(int val)` Appends the integer `val` to the stream and returns the element representing the `k`th largest element in the stream.

Example 1:

```
Input
["KthLargest", "add", "add", "add", "add", "add"]
[[3, [4, 5, 8, 2]], [3], [5], [10], [9], [4]]
Output
[null, 4, 5, 5, 8, 8]

Explanation
```

```
{
      public: priority_queue<int, vector<int> , greater<int>> pq;
int n;
KthLargest(int k, vector<int>& nums) { n= k;
for(int i =0 ;i <nums.size();i++)
{
pq.push(nums[i]);
```

Accepted Runtime: 5 ms

Case 1

Input

```
["KthLargest","add","add","add","add","add"]
```

```
[[3,[4,5,8,2]],[3],[5],[10],[9],[4]]
```

Output

```
[null,4,5,5,8,8]
```

Activate Windows
Go to Settings to activate Windows.

AIM: Last Stone Weight

You are given an array of integers stones where stones[i] is the weight of the ith stone.

We are playing a game with the stones. On each turn, we choose the heaviest two stones and smash them together. Suppose the heaviest two stones have weights x and y with x <= y. The result of this smash is:

If x == y, both stones are destroyed, and

If x != y, the stone of weight x is destroyed, and the stone of weight y has new weight y - x.

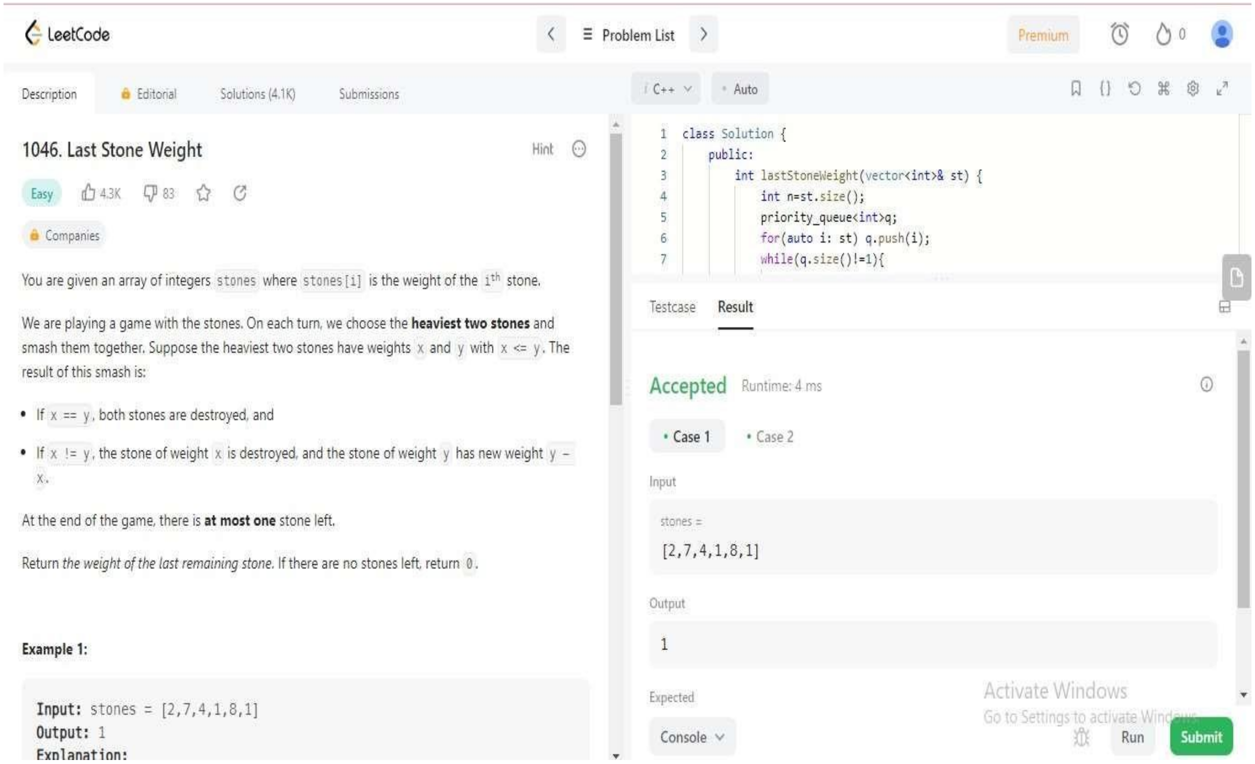
At the end of the game, there is at most one stone left.

Return the weight of the last remaining stone. If there are no stones left, return 0.

PROGRAM:

```
class Solution {
    public:
        int lastStoneWeight(vector<int>& st) {
            int n=st.size();
            priority_queue<int>
            q; for(auto i: st)
            q.push(i);
            while(q.size()!=1){
            int x=q.top();
            q.pop(); int
            y=q.top(); q.pop();
                q.push(max(x,y)-
            min(x,y)); }
            return q.top();
        }
         }
        ;
```

OUTPUT:



1046. Last Stone Weight

You are given an array of integers `stones` where `stones[i]` is the weight of the i^{th} stone.

We are playing a game with the stones. On each turn, we choose the **heaviest two stones** and smash them together. Suppose the heaviest two stones have weights `x` and `y` with `x <= y`. The result of this smash is:

- If `x == y`, both stones are destroyed, and
- If `x != y`, the stone of weight `x` is destroyed, and the stone of weight `y` has new weight `y - x`.

At the end of the game, there is **at most one** stone left.

Return *the weight of the last remaining stone*. If there are no stones left, return `0`.

Example 1:

```
Input: stones = [2,7,4,1,8,1]
Output: 1
Explanation:
```

```cpp
class Solution {
    public:
        int lastStoneWeight(vector<int>& st) {
            int n=st.size();
            priority_queue<int>q;
            for(auto i: st) q.push(i);
            while(q.size()!=1){
```

Accepted Runtime: 4 ms

Case 1 Case 2

Input

```
stones =
[2,7,4,1,8,1]
```

Output

```
1
```

Expected