Experiment 1.3

Student Name: Anant Bhardwaj **UID:** 21BCS8910
Branch: BE-CSE **Section/Group:** 21BCS_CC-648-B
Semester: 6th **Subject Code:** 21CSP-351
Subject Name: Advance Programming Lab-2
Date of performance: 02-06-2024

Aim: To demonstrate the concept of Heap model.

Objective: To understand the Heap model as a data structure and its applications, including efficient storage and retrieval of priority-based information, and to explore its implementation, operations, and complexities to facilitate effective problem-solving and algorithmic design.

PROBLEM-1: 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.

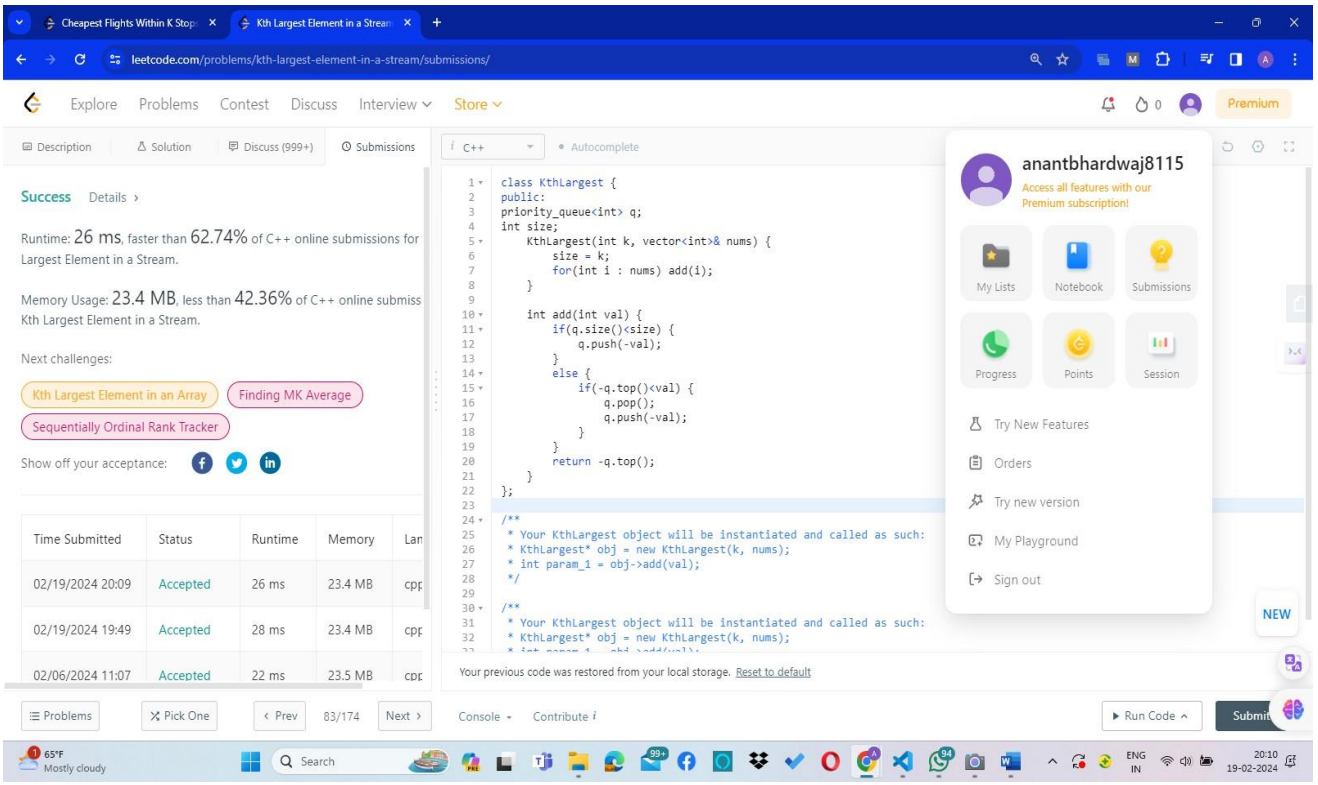
CODE:

```
class    KthLargest    {
public:
priority_queue<int> q;
int size;
   KthLargest(int k, vector<int>& nums) {
size = k;        for(int i : nums) add(i);
   }

   int   add(int   val)   {
if(q.size()<size)        {
q.push(-val);            }
```

```
else {                    if(-
q.top()<val)              {
q.pop();
            q.push(-val);
        }       }
return -q.top();
    }
};
```

OUTPUT:



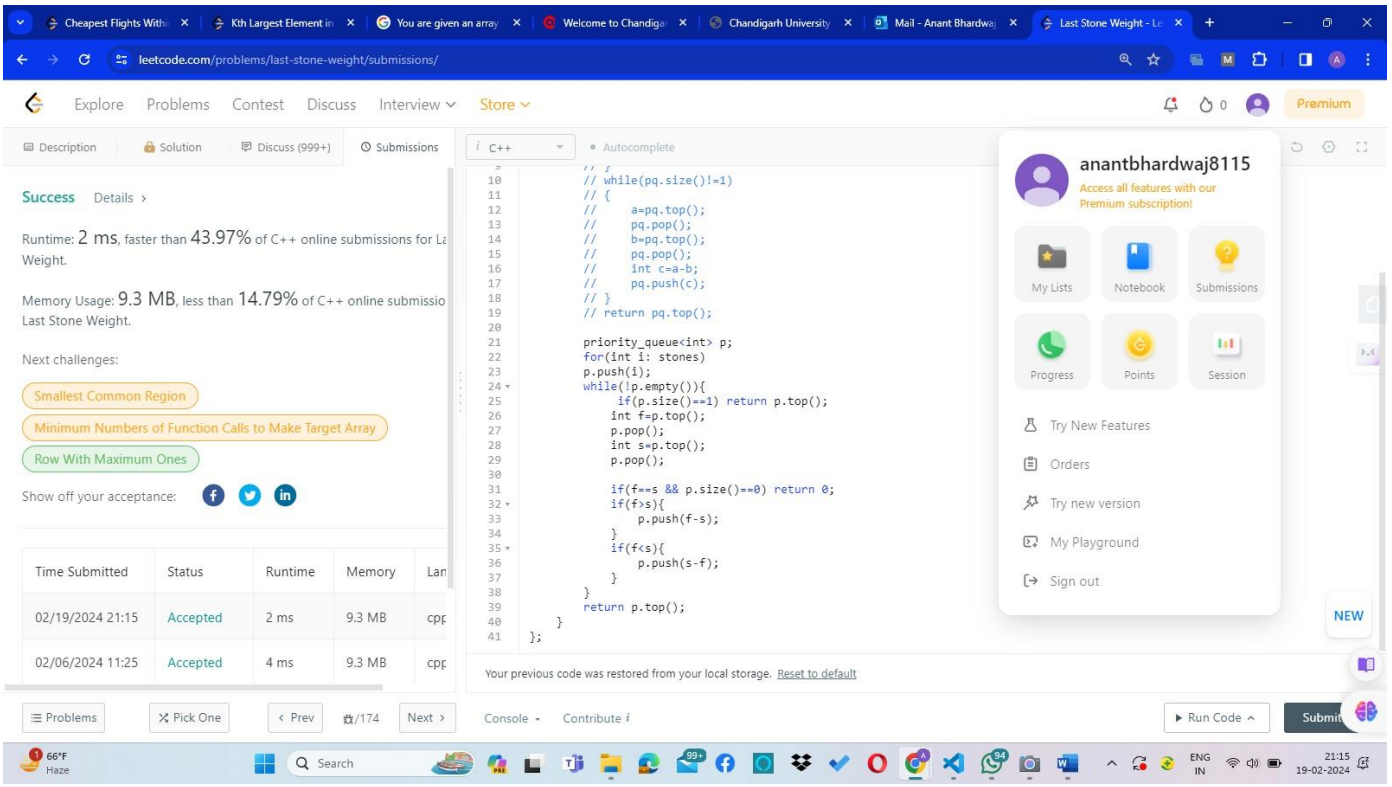
PROBLEM-2: You are given an array of integers stones where stones[i] is the weight of the ith stone. We are playing a game with the stones. On each turn, we choose the heaviest two stones and smash them together. Suppose the heaviest two stones have weights x and y with x <= y. The result of this smash 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.

CODE:

```
class Solution { public:     int
lastStoneWeight(vector<int>&
stones) {
priority_queue<int> p;
for(int i: stones)
        p.push(i);
while(!p.empty()){
if(p.size()==1) return p.top();
            int f=p.top();
            p.pop();
int s=p.top();
            p.pop();
if(f==s && p.size()==0)
return 0;
            if(f>s){
                p.push(f-s);
}             if(f<s){
                p.push(s-f);
            }         }
return p.top();
    }
};
```

OUTPUT:



Learning Outcomes:

- Understanding the properties and characteristics of a Heap, including its structure, ordering, and representation.

- Implementing Heap data structure and related algorithms in programming languages such as Java, or C++.