Experiment 6.2

Student Name: Amit Kumar **UID: 22BCS16121**
Branch: CSE **Section: 641/A**
Semester: 6th **DOP: 25/02/2025**
Subject: AP **Subject Code:22CSP-351**

Aim:

Problem : : Kth Largest Element in an Array
Problem statement: Given an integer array nums and an integer k,
return *the* kth *largest element in the array*.
Note that it is the kth largest element in the sorted order, not the kth distinct element.
Can you solve it without sorting?

Example 1:
Input: nums = [3,2,1,5,6,4], k = 2
Output: 5
Example 2:
Input: nums = [3,2,3,1,2,4,5,5,6], k = 4
Output: 4

Algorithm:

- **Initialize a Min-Heap**:
  - Create an empty **min-heap** to store the top k largest elements.
  - This will be used to efficiently retrieve the smallest of the k largest elements.
- **Iterate through each element in the array**:
  - Loop through all elements num in the input array nums.
- **Add the current element to the heap**:
  - For each num, add it to the min-heap (minHeap.push(num)).
- **Maintain the heap size of k**:
  - If the size of the heap exceeds k, remove the smallest element from the heap (minHeap.pop()).
- **Continue until all elements are processed**:
  - Repeat steps 3 and 4 for all elements in the array.
- **Retrieve the Kth largest element**:
  - After processing all elements, the root of the heap (minHeap.top()) will contain the **Kth largest element** in the array.
- **Return the Kth largest element**:
  - Return the value at the root of the min-heap.

Code:

```
#include <vector>
#include <queue>  // For priority_queue (min-heap)
using namespace std;

class Solution {
public:
    int findKthLargest(vector<int>& nums, int k) {
        // Min-heap to store the top k largest elements
        priority_queue<int, vector<int>, greater<int>> minHeap;

        // Iterate through all elements in nums
        for (int num : nums) {
            minHeap.push(num);  // Insert current number into heap

            // If the heap size exceeds k, pop the smallest element
            if (minHeap.size() > k) {
                minHeap.pop();
            }
        }

        // The root of the heap is the kth largest element
        return minHeap.top();
    }
};
```

Output:

Code

Test Result | Testcase | Note

Accepted Runtime: 0 ms

• Case 1 • Case 2

Input

nums =
[3,2,1,5,6,4]

k =
2

Output

5

Expected

5

Contribute a testcase

Code

Test Result | Testcase | Note

Accepted Runtime: 0 ms

• Case 1 • Case 2

Input

nums =
[3,2,3,1,2,4,5,5,6]

k =
4

Output

4

Expected

4

Contribute a testcase