**Infinite Champions Programme – Day 2 (Assignment Sheet)**

**Instructions  
• Deadline: Submit your solutions by 27th September, 2025, EOD.  
• Platform: Test your solutions on LeetCode  
• Collaboration: Discussing concepts is encouraged, but all code must be your own.**

**1.** [**Merge Sorted Array (88)**](https://leetcode.com/problems/merge-sorted-array/)

* **Problem:** You are given two sorted arrays, nums1 and nums2, where nums1 has extra space at the end to accommodate all elements of nums2. Merge the two arrays into a single sorted array in-place.
* **Objective:** Implement a function that merges two sorted arrays efficiently using pointers from the end.
* **YouTube Solution (Java):** [Merge Sorted Array – Java Solution](https://www.youtube.com/watch?v=8CdySAPnEGY)

class Solution {

    public void merge(int[] nums1, int m, int[] nums2, int n) {

        int[] a=new int[n+m];

        int i=0;

        int j=0;

        int k=0;

        while(i<m && j<n){

            if(nums1[i]<nums2[j]){

                a[k]=nums1[i];

                i++;

            }

            else{

                a[k]=nums2[j];

                j++;

            }

            k++;

        }

        while(i<m){

            a[k++]=nums1[i++];

        }

        while(j<n){

            a[k++]=nums2[j++];

        }

        for(int x=0;x<n+m;x++){

            nums1[x]=a[x];

        }

    }

}

**2.** [**Kth Largest Element in an Array (215)**](https://leetcode.com/problems/kth-largest-element-in-an-array/)

* **Problem:** You are given an unsorted array of integers. Find the Kth largest element in the array.
* **Objective:** Implement a function using Quickselect (or a heap) to efficiently determine the Kth largest element.
* **YouTube Solution (Java):** [Kth Largest Element in an Array – Java Solution](https://www.youtube.com/watch?v=XEmy13g1Qxc)

class Solution {

    public int findKthLargest(int[] nums, int k) {

       PriorityQueue<Integer> pq=new PriorityQueue<>();

       for(int i=0;i<nums.length;i++){

         pq.add(nums[i]);

         if(pq.size()>k){

             pq.poll();

          }

       }

       return pq.poll();

    }

}

**3.** [**Valid Perfect Square (367)**](https://leetcode.com/problems/valid-perfect-square/)

* **Problem:** You are given a positive integer num. Determine if it is a perfect square without using built-in square root functions.
* **Objective:** Implement a binary search approach on the number range to verify whether a number is a perfect square.
* **YouTube Solution (Java):** [Valid Perfect Square – Java Solution](https://www.youtube.com/watch?v=cW3fTKGT7rU)

class Solution {

    public boolean isPerfectSquare(int num) {

        if(num==1){

            return true;

        }

        int l=1;

        int r=num;

        while(l<=r){

            long mid=l+(r-l)/2;

            long s=mid\*mid;

            if(s==num){

                return true;

            }

            else if(s>num){

               r=(int)mid-1;

            }

            else{

                l=(int)mid+1;

            }

        }

        return false;

    }

}

**Submission Checklist  
• Time and space complexity analysis for each solution.  
• Test cases demonstrating the correctness of your solutions.**