# Recording

**Date** : 18/07/2025

**Problem Specification** : Find the kth largest element in an ArrayList of integers.

**Assumption** : Assume k is a positive integer and the ArrayList contains integers.

**Limitation** : This program only works with integer values. It returns -1 for empty ArrayList, k ≤ 0, or k > ArrayList size.

**Input** : ArrayList of integers (nums), Integer k

**Processing** :

1. Validate input parameters (check for null ArrayList, empty ArrayList, invalid k values)
2. Sort the ArrayList in descending order
3. Return the element at position (k-1) in the sorted ArrayList

**Output** : The kth largest element in the ArrayList, or -1 if input is invalid.

**Algorithm** :

- **Step 1:** Check if ArrayList is null, empty, or k is invalid (≤ 0 or > ArrayList size)
- **Step 2:** If invalid, return -1
- **Step 3:** Sort the ArrayList in descending order using comparator (b - a)
- **Step 4:** Return the element at index (k-1) from the sorted ArrayList
- **Step5:** End the Programme

**Programme listing** : Programme file attached

**Test data and expected output** :

- Test data: nums = [3, 2, 1, 5, 6, 4], k = 2
  Expected output: 5
- Test data: nums = [3, 2, 1, 5, 6, 4], k = 7
  Expected output: -1
- Test data: nums = [], k = 1
  Expected output: -1

**Output obtained for test data :**

- Test data: nums = [3, 2, 1, 5, 6, 4], k = 2
  Obtained output: The 2th largest element is: 5

- Test data: nums = [3, 2, 1, 5, 6, 4], k = 7
  Obtained output: The 7th largest element is: -1

- Test data: nums = [], k = 1
  Obtained output: The 1th largest element is: -1

**Analysis** : The numbers of operation required in performing the algorithm.

| | +,- | /,* | % | </>/<=/>= | Sort() |
|---|---|---|---|---|---|
| For calculation | - | - | - | 4 comparisons | O(n log n) |
| Input validation | *4* times | - | - | *4* times | - |

**Conclusion** : This program finds and returns the kth largest element from an ArrayList of integers. The solution uses input validation to handle edge cases and built-in sorting functionality for efficient implementation. Three main variables are used: nums (ArrayList), k (position), and result (return value).

**Discussion** : The time complexity of this algorithm is O(n log n) due to the sorting operation, where n is the size of the ArrayList. The space complexity is O(1) as sorting is done in-place. For better performance with large datasets, a QuickSelect algorithm could be implemented with O(n) average time complexity. The current solution handles all edge cases by returning -1 for invalid inputs, making it robust for practical use.