**Question (Problem solving)**

**Write a program to find the key element in sorted and rotated Array**

**Steps**

1. Take input of an array n.
2. sort the array in nlogn(**merge**, quick) time complexity
3. Left Rotate the array by the mid element.
4. Write a program to search an element in time complexity **O(logn)**

**Arr[] =** { 3,4,2,1,9,8,7} **//** original array

**Arr[] =** {1,2,3,4,7,8,9}  **//** sorted array

**Arr[]** = {4,7,8,9,1,2,3} // rotated array

**Key =** 8

**Output**

Key found at position 2

**-----------------------------------------------------------------------------------------------------------------**

**Question 2 (Problem Solving)**

You are given an array A consisting of N positive integers. Each integer represents the number of players from a country, for a total of N countries.

Find the maximum number of teams that can be formed by forming teams of size **K** such that each player in the team is from a different country.

**Input:** N = 4, K = 3, arr[] = {4, 3, 5, 3}

**Output:** 5

**Explanation:**

Consider the countries are named W, X, Y and Z. The possible ways of forming the teams are {W1, X1, Y1}, {W2, Y2, Z1}, {W3, X2, Y3}, {X3, Y4, Z2}, {W4, Y5, Z3} such that in each set there is no more than 1 person from a country.

Therefore, the total count of teams formed is 5.

**Solution**

package com.greatlearning.iitr.dsa.mentorsession2;

class TeamFormation {

// Function to find if T number of teams can be formed or not

public static boolean checkTeamFormationPossible(int[] teams, int T, int k) {

// Store sum of array elements

int sum = 0;

// Traverse the array

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

sum += Math.min(T, teams[i]);

}

// Required Condition

return (sum >= (T \* k));

}

// Function to find the maximum number of teams possible

public static int countOfTeams(int[] teams\_list, int N, int K) {

int lowerBound = 0;

int upperBound = 1000;

// perform Binary Search

while (lowerBound <= upperBound) {

int mid = lowerBound + (int) (upperBound - lowerBound) / 2;

if (checkTeamFormationPossible(teams\_list, mid, K)) {

if (!checkTeamFormationPossible(teams\_list, mid + 1, K)) {

return mid;

}

else {

lowerBound = mid + 1;

}

}

else {

upperBound = mid - 1;

}

}

return 0;

}

// Driver Code

public static void main(String args[]) {

int[] arr = { 4,3,5,3};

int K = 3;

int N = arr.length;

System.out.println(countOfTeams(arr, N, K));

}

}

**Learning Objectives**

1. Develop/Focus on logical skills to solve problems with efficient logic.
2. Practice DSA concepts to solve problems.
3. Developing the ability to interpret and understand requirements provided in a problem statement.