## Questions:

Given an array of n integers. The task is to find the first element that occurs k number of times. If no element occurs k times the print -1. The distribution of integer elements could be in any range.

**Examples:**

**Input**: {1, 7, 4, 3, 4, 8, 7},

k = 2

**Output**: 7

Both **7** and **4** occur 2 times.

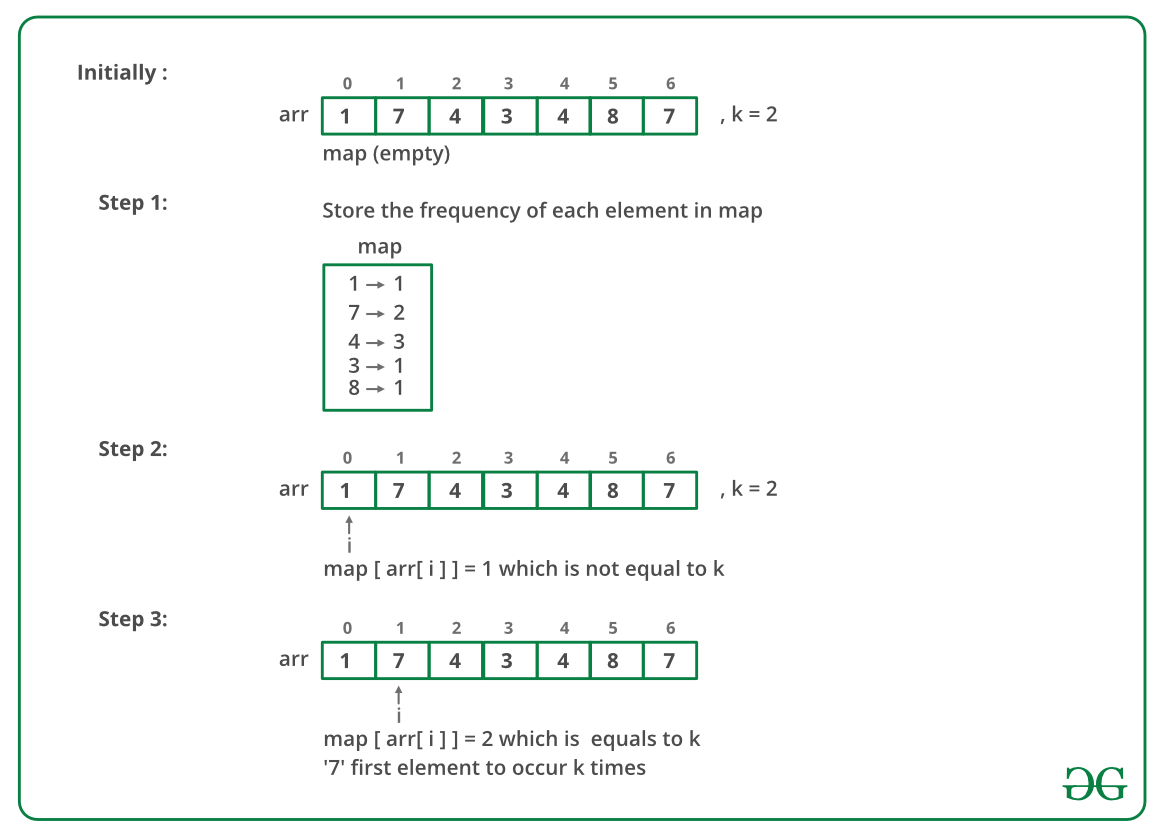
But **7** is the first that occurs 2 times.

## Solutions:

**Simple Approach:** By using two loops, count the number of times a number appears in the array.  
**Time complexity**: O(n2).  
**Efficient Approach:** Use unordered map for hashing as range is not known. Steps:

1. Traverse the array elements from left to right.
2. While traversing increment their count in the hash table.
3. Again, traverse the array from left to right and check which element has a count equal to k. Print that element and stop.
4. If no element has a count equal to k, print -1.

Below is a dry run of the above approach:



|  |
| --- |
| import java.util.HashMap;// Java implementation to find first// element occurring k timesclass GFG {// function to find the first element// occurring k number of timesstatic int firstElement(int arr[], int n, int k) {// unordered\_map to count// occurrences of each elementHashMap<Integer, Integer> count\_map = new HashMap<>();for (int i = 0; i < n; i++) {int a = 0;if(count\_map.get(arr[i])!=null){a = count\_map.get(arr[i]);}count\_map.put(arr[i], a+1);}//count\_map[arr[i]]++;for (int i = 0; i < n; i++) // if count of element == k ,then// it is the required first element{if (count\_map.get(arr[i]) == k) {return arr[i];}}// no element occurs k timesreturn -1;}// Driver program to test abovepublic static void main(String[] args) {int arr[] = {1, 7, 4, 3, 4, 8, 7};int n = arr.length;int k = 2;System.out.println(firstElement(arr, n, k));}} |

## Output:

7

**Time Complexity:** O(n)