Filter Elements ★

Points: 553.1800000000001 Rank: 1150

Problem Submissions Leaderboard

RATE THIS CHALLENGE

★ ★ ★ ★ ★

Given a list of N integers A = [a_1, a_2, ..., a_N], you have to find those integers which are repeated at least K times. In case no such element exists you have to print `-1`.

If there are multiple elements in A which are repeated at least K times, then print these elements ordered by their first occurrence in the list.

Let's say A = [4, 5, 2, 5, 4, 3, 1, 3, 4] and K = 2. Then the output is

```
4 5 3
```

because these numbers have appeared at least 2 times.

Among these numbers,

4 has appeared first at position 1,

5 has appeared next at position 2,

and 3 has appeared thereafter at position 6.

That's why, we print in the order 4, 5 and finally 3.

Input

First line contains an integer, T, the number of test cases. Then T test cases follow.

Each test case consist of two lines. First line will contain two space separated integers, N and K, where N is the size of list A, and K represents the repetition count. In the second line, there are N space separated integers which represent the elements of list A = [a_1, a_2, ..., a_N].

Output

For each test case, you have to print all those integers which have appeared in the list at least K times in the order of their first appearance, separated by space. If no such element exists, then print `-1`.

Constraints

1 <= T <= 10

1 <= N <= 10000

1 <= K <= N

1 <= a_i <= 10^9

Sample Input

```
3
9 2
4 5 2 5 4 3 1 3 4
9 4
4 5 2 5 4 3 1 3 4
10 2
5 4 3 2 1 1 2 3 4 5
```

Sample Output

```
4 5 3
-1
5 4 3 2 1
```

Explanation

Sample Case #01: This is the same example mentioned in the problem statement above.

Sample Case #02: As no elements repeats more than 3 times, we don't have any elements satisfying the criteria of minimum K times.

Sample Case #03: All elements are repeated 2 times. So we print all of them according to their order of occurance, which is 5 -> 4 -> 3 -> 2 -> 1.

Change Theme Language Haskell

```haskell
import Data.Map.Strict as M
import Data.List as L
import Control.Monad


filterElem :: [Int] -> Int -> [Int]
filterElem xs k = L.filter (flip M.member (M.filter (>= k) m)) $ L.nub xs
    where m = L.foldl' (\mp elem -> M.insertWith (+) elem 1 mp) M.empty xs

main = do
    n <- read <$> getLine
    forM [1..n] $ const $ do
        [n', k] <- (fmap read . words) <$> getLine
        xs <- (fmap read . words) <$> getLine
        case filterElem xs k of
          [] -> putStrLn "-1"
          xs' -> putStrLn $ unwords $ fmap show xs'


```

Line: 19 Col: 1

Upload Code as File Test against custom input Run Code Submit Code