Word Counting 2

> “I have hated words and I have loved them, and I hope I have made them right.” — Markus Zusak

We also love words, don’t we? However, we are code poets, so, we love words in a different way. How about counting them? Yes, counting word seems to be a good thing and has a huge application in computer science. Well, counting may sometimes sound boring, but many hidden knowledge can be achieved from counting. For example, finding the most frequent word in vast collection of documents, which then in turns, can be used to classify those documents.

But fear not! We are not interested in large documents right now. In fact, in our extremely busy world, offline data processing algorithms are getting obsolete day by day. Instead of words in large files, we are now more interested in streams of words.

The idea is quite simple, your program will keep reading from a stream where new words are continuously being fed to your program, but due to limited storage, your program can only remember the latest K words. So, when $(K+1)^{th}$ word arrives, your program forgets the 1^{st} word, when $(k+2)^{th}$ word arrives, your program forgets the 2^{nd} word, and so on.

We want you to find the most frequent word over the latest K words each time a new word arrives.

Input Format

The first line of input will contain an integer T, which denotes the number of test cases. First line of each test case will contain two integers N and K, where N is the total number of words to be read, and K is the number of words that your program can remember. Each of the following N lines will contain a single word composed with only lowercase characters ‘a’ to ‘z’, at most 8 characters long.

Output Format

For each case, first print the case number, then for each of the words

For each case, first print the case number, then for each of the words, print the most frequent word that appears within the last K words along with the frequency. If there are less than K words, then print the most frequent word among all of them. In case there is a tie, print the alphabetically smaller word. See sample input and output sections for more details.

Constraints

$1 \leq T \leq 20$

$1 \leq K \leq N \leq 10^5$

$1 \leq$ word-length ≤ 8

Sample Input

```
1
8 3
hello
hi
who
hi
hi
hello
who
when
```

Sample Output

```
Case 1:
hello 1
hello 1
hello 1
hi 2
hi 2
hi 2
hello 1
hello 1
```

Explanation

The following table shows a simulation of the program for the entry of each new word. Note, once the capacity of the program is full, i.e. equals to K, the program will start forgetting the oldest entry for each

new entry. In other words, those words will not be taken into consideration anymore.

#	New word in stream	Words remembered by the program	Most frequent word (Count)
1	hello	hello	hello (1)
2	hi	hello, hi	hello (1)
3	who	hello, hi, who	hello (1)
4	hi	hi, who, hi	hi (2)
5	hi	who, hi, hi	hi (2)
6	hello	hi, hi, hello	hi (2)
7	who	hi, hello, who	hello (1)
8	when	hello, who, when	hello (1)