Problem A - Eficiência no Teclado

Check our instructions page for detailed information on the qualification and the format of this problem.

Laura, a former ONI participant, recently entered university and plans to take part in university programming competitions as a team, along with other former ONI competitors. In these competitions, not only are there many more problems than in ONI, but if two teams do the same number of problems, the speed with which they solve them is a decisive factor in the tie-breaker, rather than simply drawing like in ONI. To maximise team speed, Laura wants to try out some optimisations that few teams do, to ensure that her team isn't at a disadvantage and can program as quickly as possible. To do this, they started by testing various keyboards to see which one the team does best with, but they ran into some obstacles and decided to ask for your help.

Part I

To test each keyboard, they decided to write the name of $\boldsymbol{N}$ variables, each of length a_i, in this case all lowercase strings, and although they found some keyboards to be very fast, they encountered a problem: some of the keyboards consistently capitalise some of the letters when they're not supposed to. Laura wants to find out which keys are faulty and need to be fixed. Your goal, to help Laura and her team, is given a list of $\boldsymbol{N}$ variables, determine which letters appear wrongly capitalised.

Example

For example, Laura's team tried to write the variables `counter`, `iterator`, `max`, `min`, `i`, but the screen showed `cOnTadOr`, `ITeradOr`, `MaX` `MIn` and `I`, so the letters $\{i,m,o,t,x\}$ appeared capitalised when they shouldn't have, while the letters $\{a,c,d,e,n,r\}$ appeared on the screen as they were supposed to. Note that in the input of this problem, no letter will appear in uppercase and lowercase simultaneously.

Constraints

The following limits are guaranteed for all test cases of this Part that will be given to the program:

$1 \leq \boldsymbol{N} \leq 10^5$	Number of variables
$1 \leq a_1 + a_2 + \ldots + a_n \leq 10^5$	The sum of the variable lengths

It is guaranteed that in the input of Part I all the letters of the alphabet appear at least once, and that no letter appears in both lower and upper case.

The test cases in this part of the problem are organised in a single group:

Subtask	**Points**	**Additional Constraints**
1	30	No further constraints

Part II

After getting what they think is their best keyboard, they run into another problem: they don't know what variable format to use. Not only do they not agree with each other, they're not always consistent in their own code. But in this type of competition, it's common to have a team-mate helping with the writing and especially the debugging of the code, so they want to make sure they're consistent in the variable format they use in the training sessions so they get used to it and then it's much easier to read each other's code on the day of the competition.

The types of variables used by team members in old code can be summarised as the following **4** types:

1. *PascalCase*: variables composed of several words where each word has its initial capitalised.
2. *dromedaryCase*: variables composed of several words where each word, except the first, has its initial capitalised.
3. *snake_case*: variables composed of several words where these are divided by underscores '_'.
4. *kebab-case*: variables composed of several words where they are divided by hyphens '-'.

They've decided to compile a list of the $\boldsymbol{N}$ variables made up of several words, of total length a_i, that they've used in the past, and they want your help to quickly determine the format of each one so that they can then decide which format to use.

So, given a list of $\boldsymbol{N}$ variables, determine the format of each one.

Example

If the list contains the variables `mid-point`, `AdjacencyList`, `max_flow`, `edgeCase`, `NodePai`, `olimpiadas-de-informatica` you should print them, respectively, *kebab-case*, *PascalCase*, *snake_case*, *dromedaryCase*, *PascalCase* and *kebab-case*.

Note that in this Part it is guaranteed that all variables will be of one of the 4 types {PascalCase, dromedaryCase, snake_case, kebab-case}, i.e. there will be no single-word variables to avoid ambiguity, and there will be no variables that "mix" types such as `pastel-de_nata` or `Department-of-Computer-Science`.

Constraints

The following limits are guaranteed for all test cases of this Part that will be given to the program:

$1 \leq \boldsymbol{N} \leq 10^5$	Number of variables
$1 \leq a_1 + a_2 + \ldots + a_n \leq 10^5$	The sum of the variable lengths

The test cases in this part of the problem are organised in a single group:

Subtask	Points	Additional Constraints
2	30	No further constraints

Part III

After carefully analysing your list, they decided that they were going to use the *dromedaryCase* format for all their variables. However, Laura had already started writing a new programme and it already had some variables in other formats. Laura then gave you a list of the $\boldsymbol{N}$ variables and asked you to convert them to the *dromedaryCase* format.

In this part, single-word variables count as *dromedaryCase* if they only consist of lowercase letters and as *PascalCase* if the first letter is capitalised.

Example

If the list contains the variables `mid-point`, `AdjacencyList`, `max_flow`, `Grid`, `edgeCase`, `NodeFather`, `computer-optimised` you must print them, respectively, `midPoint`, `AdjacencyList`, `maxFlow`, `grid`, `edgeCase`, `nodePai` and `olimpiadasDeInformatica`.

Note that in this Part it is guaranteed that all variables will be of one of the 4 types {PascalCase, dromedaryCase, snake_case, kebab-case}, i.e. there will be no single-word variables to avoid ambiguity, and there will be no variables that "mix" types such as `pastel-de_nata` or `Department-of-Computer-Science`.

Constraints

The following limits are guaranteed for all test cases of this Part that will be given to the program:

$1 \leq \boldsymbol{N} \leq 10^5$	Number of variables
$1 \leq a_1 + a_2 + \ldots + a_n \leq 10^5$	The sum of the variable lengths

The test cases in this part of the problem are organised in a single group:

Subtask	Points	Additional Constraints
3	40	No further constraints

Summary of Subtasks

The test cases for the problem are organized into three groups with different additional restrictions:

Subtask	Points	Part	Additional Constraints
1	30	Part I	No further constraints
2	30	Part II	No further constraints
3	40	Part III	No further constraints

Input Format

The first line contains an integer $\boldsymbol{P}$, which represents the Part of the test case. If this is 1, then the test case refers to Part I, if it is 2, then it refers to Part II and if it is 3, then it refers to Part III.

This is followed by a line with an integer $\boldsymbol{N}$, the number of variables in the list.

Finally, there are $\boldsymbol{N}$ lines, each with the name of a variable.

Output Format

In Part I, the output must contain only one line, made up of all the letters that were incorrectly capitalised in the input separated by spaces, in alphabetical order and in **lowercase**.

In Part II, the output must contain $\boldsymbol{N}$ lines, where the i-th line indicates the format (*PascalCase*, *dromedaryCase*, *kebab-case* or *snake_case*) of the i-th word.

In Part III, the output must contain $\boldsymbol{N}$ lines, where the i-th line contains the i-th variable converted to the format *dromedaryCase*

Note: there should be no space at the end of each line (i.e. after each letter/string there should only be one line change). If this format is not respected, the result of a submission will be `Presentation Error` (see the instructions for more information).

Example 1 Input

```
1
12
ArrOz
blOcO
ItErAdOr
fAcE
grElhA
jOkEr
mAx
nOdE
pAI
qUIcksOrt
vErtIcE
why
```

Example 1 Output

```
a e i o u
```

Example 2 Input

```
2
6
mid-point
ListaDeAdjacencia
max_flow
edgeCase
NodePai
olimpiadas-de-informatica
```

Example 2 Output

```
kebab-case
PascalCase
snake_case
dromedaryCase
PascalCase
kebab-case
```

Example 3 Input

```
3
7
mid-point
ListaDeAdjacencia
max_flow
Grelha
edgeCase
NodePai
olimpiadas-de-informatica
```

Example 3 Output

```
midPoint
listaDeAdjacencia
maxFlow
grelha
edgeCase
nodePai
olimpiadasDeInformatica
```