F - Prefetching

Context

Processors are more and more powerful, being able to run in a short time algorithms that would have taken months in old processors. They are currently able to execute hundreds of instructions simultaneously. But those instructions may require data stored in memory, a much slower device than the processor. Computer architects have struggled to solve this problem and have found a good solution: prefetching.

The Problem

Prefetching consists in predicting the data that will be required by the processor and request it in advance. Usually, the same instruction accesses the same data, consecutive data, or data separated by the same distance. For example, let us assume that instruction "a" accesses the data located in the position 1, then in a new execution accesses the data located in the position 4, and then in the position 7. The distance between two accesses is 3 (4 - 1 = 3 and 7 - 4 = 3). Note that distances can be negative too.

Your task is to compute, for each instruction accessing memory in the program, the distance that should be selected to predict successfully the next address accessed by the same instruction more times. In case that more than one distance predicts the maximum number of addresses, the lower in absolute number should be selected. In case of a draw in absolute number, the positive one should be selected.

The Input

Your input consists of several programs separated by a line with the character '.'. The last program ends with a line with the character '#'. Each program consists of up to 10,000 instructions accessing memory. Each line in the program represents an instruction executed, and it consists of a lower-case letter identifying the instruction and a positive integer M ($0 \leq M < 2^{30}$) indicating the location of the data in memory.

The Output

For each program, you should compute the following. For each instruction in the program, and in the order of its first appearance in the trace, your program must print the letter of the instruction, the number of executions of that instruction, the best distance selected, and the number of good predictions that it will achieve. In case of a single occurrence of an instruction, neither distance nor good predictions should be printed. Outputs of different programs are separated by a line containing '.' and after the output for the last program a line containing '#' should appear.

Sample Input

```
a 1
c 400
b 20
a 4
a 7
b 10
a 9
.
```

```
z 10000
y 20000
z 20000
z 10000
#
```

Sample Output

```
a 4 3 2
c 1
b 2 -10 1
.
z 3 10000 1
y 1
#
```