G - Prefetching, again

Background

Bringing close to the processor the data needed to perform computations ahead of time is key for high-performance computers. This is the purpose of data prefetching, which requires a prediction of the future memory addresses that the processor will request. In the last edition of the OMP, we set a prefetching problem that required to compute *strides*. In a few days, a new prefetching technique, born at the University of Murcia, will be disclosed worldwide. This new prefetcher is based on *deltas* instead of *strides* and, quoting a famous advertising, we can say that is *Probably the Best Prefetcher in the World*.

The Problem

Applications have a set of memory instructions. Each of these instructions accesses a memory location each time it is executed. We define a delta as the difference between the currently accessed location and one of the previous accessed locations. For example, let's assume that instruction "a" accesses the data located in the memory address 1. Then, in a new execution, instruction "a" accesses the data located in the position 3, and then, in the position 6. When accessing position 6, the two computed deltas are 3 (6 - 3 = 3) and 5 (6 - 1 = 5). Note that deltas can be negative too.

Your task is to compute, for each instruction accessing memory in the program, the delta that should be selected to predict successfully one of the 3 subsequent addresses accessed by the same instruction more times. To compute the delta, you have to check the last 3 executions of the same instruction. If the delta is 0, that delta should not be considered (but it accounts in the search of the last 3 executions). In case that more than one delta 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 memory accesses. Each line in the program represents a memory access, and it consist of a lower-case letter identifying the instruction and a positive integer M ($0 \leq M < 2^{30}$) indicating the accessed location.

The Output

For each program you should print the following output. For each instruction in the program, and in the order of its first appearance in the program, you must print the letter of the instruction, the number of times that instruction was executed, the best delta (considering the next 3 executions) for that instruction, and the number of times that the delta will predict correctly any of the next 3 executions. In case of a single occurrence of an instruction, neither delta not 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 3
a 6
b 10
a 8
a 11
.
z 10000
y 20000
z 20000
z 10000
#
```

Sample Output

```
a 5 5 3
c 1
b 2 -10 1
.
z 3 10000 1
y 1
#
```

OMP'22
Facultad de Informática
Universidad de Murcia