Problem 3
Preprossing

MIDI (Musical Instrument Digital Interface) is a standard for communication involving computers and synthesized music instruments. Part of the standard defines commands, which when transmitted to a synthesizer, begin and end the sounding of a particular note. In this problem we will consider processing simple MIDI "programs." In the following example, three simultaneous notes (a chord, with note numbers 60, 70 and 80) are played for 10 time units immediately followed by a single note (number 62) for 2 time units.

```
0 ON 60
0 ON 70
0 ON 80
10 OFF 60
10 OFF 80
10 OFF 70
10 ON 62
12 OFF 62
```

Much existing music cannot be directly translated to this program form. Sometimes a note is already "on" when the written music indicates that it is to be sounded again. For example:

```
0 ON 60
10 ON 60
12 OFF 60
20 OFF 60
```

A synthesizer will interpret this program to sound note 60 for 12 time units, not 20 as indicated. We will not hear the separate sounding of the note at time 10, since turning on a note that is already sounding will be ignored. By analogy, consider turning a light on and off. If it's on, turning it on again is ineffective. Likewise, the first time that a light is turned off, it is off!

When a note already on is to be sounded again, the program can be "fixed" by inserting an OFF command for that note 1 time unit before the second ON command. Since there are already at least two OFF commands in such circumstances, only the last of these should be retained; the other should be eliminated from the program. The "fixed" program will cause the synthesizer to behave as if the same note had been played twice in rapid succession.

Another problem exists in programs that turn a note on and off at the same time. Depending on the ordering of the events in the program, either the note will be prematurely ended (if the OFF command appears after the ON), or the second sounding of the note will not be heard. For example:

```
0 ON 60           0 ON 60
10 ON 60          10 OFF 60
10 OFF 60         10 ON 60
20 OFF 60         20 OFF 60
```

In the example on the left, the note will be turned off at time 10. In the example on the right, the note isn't left off long enough to allow a human listener to detect the "punctuation" in the sound. In both cases the correction is the same: move the OFF command so it is executed by the synthesizer 1 time unit before the corresponding ON command. If an OFF command inserted 1 time unit before an ON as a result of the "fix" occurs at exactly the same time as the preceding ON, the second ON and the OFF that occurs at the same time should be eliminated. Write a program that will accept an arbitrary number of MIDI programs and "fix" them as described above.

Input

Each program contains an arbitrary number of lines. Each line contains, in order, the time that the command is sent to the synthesizer (a non-negative integer), a command (either ON or OFF), and a note (an integer in the range 1 to 127). These items are separated by one or more blanks. The last line of each program will be the integer -1.

Output

The output is to be a "fixed" MIDI program in the same format as the input.

Notes

- The ON and OFF commands will always be in upper case letters.
- The times associated with programs are in non-decreasing order.
- All notes are initially OFF.
- If different notes are to be turned on or off simultaneously, the order in which the corresponding commands appear is unimportant.
- Each ON command will have a matching OFF command following it in the program.

Sample input

```
file.txt
```

Sample input file (file.txt)

```
0 ON 60
10 ON 60
12 OFF 60
20 OFF 60
-1
```

Sample output

```
0 ON 60
9 OFF 60
10 ON 60
20 OFF 60
```

Sample input file (file.txt)

```
0 ON 60
5 ON 70
10 ON 60
10 OFF 60
15 OFF 70
15 ON 70
20 OFF 60
20 OFF 70
-1
```

Sample output

```
0 ON 60
5 ON 70
9 OFF 60
10 ON 60
14 OFF 70
15 ON 70
20 OFF 60
20 OFF 70
```

Sample input file (file.txt)

```
0 ON 60
1 OFF 60
1 ON 60
10 OFF 60
-1
```

Sample output

```
0 ON 60
10 OFF 60
```