You are reading data from a zombie sensor. The sensor scans the area to obtain the number of zombies in the immediate area. The zombie sensor normally writes log entries in the form of "Zombies: <integer>" or "No Zombies:" to its buffer as it performs scans, but it may also write "RUN," when the sensor is overloaded. These are the only values that will be written to the buffer.

The zombie sensor’s serial port emits a line containing whatever data is in its buffer every second, regardless of whether the buffer contains a complete log entry, or even multiple entries.

A valid sequence of log entries may be:

```
Zombies: 5;
Zombies: 1;
No Zombies;
Zombies: 70;
RUN;
RUN;
```

But the sensor’s serial port may emit:

```
Zombies: 5;Zombies: 1;
No Zombies;
Zombies 70;
RUN;
RUN;RUN;Zo
```

It is imperative to process the serial port data correctly if you are to survive.

**Input**

The first line of input contains the number of data sets, $N$ (1 ≤ $N$ ≤ 50). For each data set, the input contains the raw data emitted by the zombie sensor’s serial port (see above for details) followed by a line containing only the string ‘END OF CASE’. Since data is emitted by the zombie sensor’s serial port once per second, the first line of input is read after 1 second, the 2nd line after 2 seconds, and so on.

**Output**

For each complete log entry, you should output a line containing ‘timestamp: log_entry’, where timestamp is the number of seconds elapsed between the start of the data set and the time at which the entry was completely parsed.

**Sample Input**

```
2
Zombies: 5;
Zombies: 1;
No Zombies;
Zombies: 70;
RUN;
RUN;
END OF CASE
No Zombies;
No Zombies;
Zombies: 4;Zombies: 14;
Zombies: 60;
Zombies: 100;
Zombies: 15;
RUN;
RUN;RUN;
R
END OF CASE
```

**Sample Output**

```
3: Zombies: 5;
4: Zombies: 1;
5: No Zombies;
6: Zombies: 70;
7: RUN;
8: RUN;
9: RUN;
2: No Zombies;
4: No Zombies;
5: Zombies: 4;
7: Zombies: 14;
9: Zombies: 60;
11: Zombies: 100;
14: Zombies: 15;
15: RUN;
16: RUN;
16: RUN;
```