Problem : Calculating Flight Duration

Naaah! Yet another time you receive a flight ticket and have to calculate how long your flight takes using confusing format that the airlines use to describe the departure/arrival times. But this time, you decide to solve this problem once and for all by writing a program to calculate the flight duration! For simplicity, in the first version of the problem you only handle non-stop flights.

Input (Standard Input)

The input contains several test cases. Each test case describes a non-stop flight on a single line of the input. Each flight description begins with flight number (a string of at least 2 at most 7 alphanumeric characters) followed by the departing city name, time zone, and the departure time and also the arriving city name, time zone and the arrival time. City names are strings of at least 1 at most 9 alphanumeric characters.

The time zone is written as a number inside a pair of parentheses with the format shown in the sample input, like `(+3.5)`. Note that the plus/minus sign is always present. This number specifies the offset from the UTC standard time, expressed in hours (so a time in the time zone +3.5 is three hours and half more than the time in UTC). There is no space character between the city names and the time zones.

The flight times are described in the format shown in the sample input, e.g., `4:10a`. Note that the time is presented in 12-hour format, the minutes is a two-digit number. A character ‘a’ or ‘p’ immediately follows the time indicating AM and PM times respectively. In case a flight arrives the next day, a ‘+1’ is appended to the arrival time. Assume that no flight takes more than 24 hours (hence we do not have a +2 symbol in arrival times), and there is no test case in which the flight arrives the day before (hence we do not have a -1 symbol in arrival times).

The end of input is specified by a single line containing a single # character which should not be processed.

Output (Standard Output)

For each test case, output a single line containing the flight number, one space character, and the duration of the flight. The duration must be of form <hours>:<minutes>, where <minutes> is formatted as a two-digit number.

Sample Input and Output

Standard Input	Standard Output

`EK976 Tehran(+3.5) 4:10a Dubai(+4) 6:45a`
`EK432 Dubai(+4) 8:40a Singapore(+8) 7:40p`
`JL710 Singapore(+8) 11:20p Tokyo(+9) 7:00a+1`
`EK0215 Dubai(+4) 8:20a LA(-8) 12:50p`
`#`
	`EK976 2:05`
	`EK432 7:00`
	`JL710 6:40`
	`EK0215 16:30`

Problem : Computerizing a Stockroom

Jobs is a stock clerk who has been working in a computer manufacturing company since its establishment time. He is responsible for any transaction done in the stockroom where computer parts are stored. Jobs is an old-fashioned employee who yet insists to register any transaction in his booklet and resists to make transaction registration computerized. He did not believe in modern technologies till today when his boss asked him to provide a summary report on the current state of the stockroom, including the computers given out to employees, and working and non-working computer parts remaining in the stockroom. He is so nervous as the report is due tonight. He does not have enough time to manually prepare a report from many transactions in his booklet. Now, he believes in modern technologies and asks you to write a program which gets all transactions as input, and outputs the report soon.

Each transaction written on his booklet starts with a date-time, and is followed by one of the following templates:

- Bought <*NUM*> <*PIECE*>.
- Assembled a computer for <*PERSON*> using <*PIECES*>.
- Got the computer of <*PERSON*> back and disassembled it.
- Found that <*A*> <*PIECE*> is not working.
- <*A*> <*PIECE*> is repaired and now can be used again.

The placeholders used above are defined as below (the first letter is capitalized in the case of appearing at the beginning of a sentence):

- <*A*>: it is either “a” or “an” depending on its successive word.
- <*NUM*>: it is either <*A*> (which means 1), or “X items of” where X is an integer greater than 1.
- <*PERSON*>: it represents the full name of an employee, containing a number of space-separated words all starting with capital letters.
- <*PIECES*>: it is a list of “<*NUM*> <*PIECE*>”, separated with commas. There are always at least 2 entries in the list, and an extra “and” will follow the last comma (before the last entry).
- <*PIECE*>: it is a phrase including a computer-part name (like RAM or CPU) and possibly some extra data which may represent the model, speed, capacity, etc.

You can assume that each entity (computer-part type or employee name) is always referenced with a unique case-sensitive phrase, and that no two different entities are referenced with the same phrase, even case-insensitively. All computer parts that are bought are initially working; only working computer parts are used for assembling a computer. Each employee has at most one computer at any time and all transactions are logically valid at the time of writing.

Input (Standard Input)

There are multiple test cases in the input. Each test case starts with a line containing the integer n ($1 \leq n \leq 500$), the number of transactions. A transaction is given on each of the next n lines. The date-time format is “*year*-*month*-*day hour*” where parameters *year*, *month*, *day*, and *hour* are in the ranges $[2000, 2012]$, $[1, 12]$, $[1, 31]$, and $[0, 23]$ respectively and may be padded with a “0” in the case of being less than 10. The date-times are unique within each test case. The date-time is separated from the transaction sentence with the string “ - ”. All numbers used in <*NUM*> are less than 10^5. For your convenience, each two consecutive words in the input are separated with a single space, and employee names and computer-part types are wrapped with quotation marks and all characters appearing in the input are alphanumeric. Refer to the sample input for the details. The input terminates with a line containing a single “`0`” which should not be processed as a test case.

Output (Standard Output)

For each test case, write several lines of information. On the first line, write X, the number of employees who currently have a computer in the following format:

- If $X > 1$, write “`There are X employees who currently have a computer:`”.
- If $X = 1$, write “`There is one employee who currently has a computer:`”.
- If $X = 0$, write “`No computer is currently given out to the employees.`”.

Then, print X lines, each with the full name of one of those employees. These X lines must be sorted in the lexicographic order.

On the next lines, print some sentences about the current state of all computer-part types written in the booklet. There must be exactly one line for each type. The lines must be sorted in the lexicographic order based on the piece phrases. For each piece phrase <*PIECE*>, its corresponding line must be in one of the following forms:

- If there is no item of the part type currently remaining in the stockroom, then write “There is no "<*PIECE*>" left in the stockroom.”.
- If there is only one item of the part type remaining in the stockroom, and it is working, then write “There is one "<*PIECE*>" left in the stockroom which is working.”.
- If there is only one item of the part type remaining in the stockroom, and it is not working, then write “There is one "<*PIECE*>" left in the stockroom which is not working.”.
- If there are X ($X > 1$) items of the part type left in the stockroom, and all of them are working, then write “There are X items of "<*PIECE*>" left in the stockroom, all working.”.
- If there are X ($X > 1$) items of the part type left in the stockroom, and all of them are not working, then write “There are X items of "<*PIECE*>" left in the stockroom, all not working.”.
- If there are X ($X > 1$) items of the part type left in the stockroom, and Y number of them are working and Z number of them are not working ($Y, Z > 0$), then write “There are X items of "<*PIECE*>" left in the stockroom, Y working and Z not working.”.

Print a line containing “###” between every two consecutive test cases. Note that when you compare two multi-word phrases, you first compare the first words of the two phrases. If they are equal, you must continue with the second words of the two phrases and so on in the case of equality of the second words. If all words of the shorter phrase match the corresponding words of the longer phrase, the shorter phrase comes first. In lexicographic comparison of the words, digits have a higher priority than letters, and the letters are compared case-insensitively. So, “A AB” < “A10” < “A2” < “aa” = “AA” < “AA B”.

Sample Input and Output

Standard Input

```
6
2011-3-18 12 - Bought 3 items of "CPU Pentium IV 2GHz".
2011-11-1 10 - Found that a "1GB RAM" is not working.
2012-1-20 11 - Bought an "optical mouse".
2011-11-21 15 - A "CPU Pentium IV 2GHz" is repaired and now can be used again.
2011-03-18 9 - Bought 2 items of "1GB RAM".
2011-10-18 14 - Found that a "CPU Pentium IV 2GHz" is not working.
16
2012-2-1 08 - Bought 3 items of "motherboard".
2012-2-1 09 - Bought 3 items of "Green case".
2012-2-1 10 - Bought 3 items of "dual core CPU".
2012-2-1 11 - Bought 3 items of "keyboard".
2012-2-1 12 - Bought 2 items of "optical mouse".
2012-2-1 13 - Bought 4 items of "2GB DDR3 RAM".
2012-2-1 14 - Bought 4 items of "500GB Hard".
2012-2-1 15 - Bought a "DVD Drive".
2012-2-2 09 - Assembled a computer for "Abbaas" using a "motherboard", and a "Green case".
2012-2-2 10 - Assembled a computer for "Dehghaan Fadaakaar" using a "motherboard", a "Green
   case", a "dual core CPU", a "keyboard", an "optical mouse", 2 items of "2GB DDR3 RAM", a
   "DVD Drive", and a "500GB Hard".
2012-2-2 11 - Assembled a computer for "Kokab Khaanum" using a "motherboard", a "Green
   case", a "dual core CPU", a "keyboard", an "optical mouse", 2 items of "2GB DDR3 RAM",
   and a "500GB Hard".
2012-2-3 10 - Got the computer of "Kokab Khaanum" back and disassembled it.
2012-2-4 10 - Found that a "motherboard" is not working.
2012-2-4 11 - Found that a "2GB DDR3 RAM" is not working.
2012-2-4 12 - Found that a "2GB DDR3 RAM" is not working.
2012-2-4 13 - Found that a "500GB Hard" is not working.
0
```

Standard Output

```
No computer is currently given out to the employees.
There are 2 items of "1GB RAM" left in the stockroom, 1 working and 1 not working.
There are 3 items of "CPU Pentium IV 2GHz" left in the stockroom, all working.
There is one "optical mouse" left in the stockroom which is working.
###
There are 2 employees who currently have a computer:
Abbaas
Dehghaan Fadaakaar
There are 2 items of "2GB DDR3 RAM" left in the stockroom, all not working.
There are 3 items of "500GB Hard" left in the stockroom, 2 working and 1 not working.
There are 2 items of "dual core CPU" left in the stockroom, all working.
There is no "DVD Drive" left in the stockroom.
There is one "Green case" left in the stockroom which is working.
There are 2 items of "keyboard" left in the stockroom, all working.
There is one "motherboard" left in the stockroom which is not working.
There is one "optical mouse" left in the stockroom which is working.
```

Problem : SMS Poll

The popular TV show, "90%", is conducting a live SMS poll every week. Each poll simply consists of a question followed by k choices, numbered form 1 to k. The audiences are asked to vote to the choices by sending a number from 1 to k to the program's phone number via SMS. The poll statistics is shown instantly during the program.

The Parliament has recently got suspicious over the statistics broadcasted by this TV show last week, and has hired a committee to investigate the issue. The committee has obtained the list of all SMSs sent to the last program of 90%, and is going to process the data for investigation. The poll under investigation has only **four choices**.

Since the number of SMSs is pretty high, the committee is asking you to write a computer program to extract the poll statistics from the SMS raw data. The data is provided to you as a list of phone:content pairs, where phone is the sender's phone number, and content is the SMS content, presumably containing the sender's vote. Your task is to compute the percentage of votes for each of the choices, and report it to the committee. There are some points to be considered in processing data:

- In its most general form, a phone number is of the form "A B C", where A is the country code, B is the area code, and C is the local number. For example, in +98 (21) 6616-6645, thecountry code is 98, the area code is 21, and the local number is 6616-6645. None of A, B, and C can start with a 0 digit.
- The country code is optional, and is always preceded by a + sign. If the country code is not given, it is considered as 98. The country code consists of at most 3 digits.
- If the country code is given, the number must also contain an area code. However, if the country code is omitted, the area code is optional. But if the area code is provided, it must be preceded by either a 0 digit or it is surrounded by a pair of parentheses (for example, either 02166166600 or (21)66166600). If the area code is not given, it must be considered as 21. The area code consists of at most 3 digits.
- Local numbers can have various lengths, from 3 to 8 digits. Examples are 6616, and 66166600.
- A phone number may have been recorded in various formats. For example, 09128122190, +98(912)812-2190, and 0912-812-2190 all refer to the same phone number. Here are the format rules:
    - The area code can be optionally surrounded by a pair of parentheses.
    - There might be a dash (-) between any two digits in the local number or between two parts of the three parts namely the country code, the area code, and the local number. A dash and a parenthesis can't be adjacent.
- You can assume that all phone numbers in the raw data are valid and strictly follow the rules specified above. Moreover, you can assume that any phone number in the world has a unique complete form. For example, we cannot simultaneously have two different numbers (218)4460 and (21)84460, as they both have the same complete form +982184460.
- The only valid content for an SMS is a number from 1 to 4 (not surrounded by white spaces). Any other content, such as letters and + sign, makes the SMS invalid, and such SMSs must be discarded.
- A sender might have sent several SMSs from a single phone number. In this case, only the first valid SMS must be considered, and all others must be discarded.
- Discarded SMSs must not be included in the total number and in the percentages.

Input (Standard Input)

There are multiple test cases in the input. The first line of each test case contains a positive integers n $(1 \leq n \leq 10{,}000)$, which indicates the number of SMSs in the list. The next n lines, each contains a pair $a{:}b$, where a is a phone number, and b is the SMS content. The content of each SMS is at most 30 characters where each character is an alphanumeric character or belongs to the set $\{\text{“}:\text{”}, \text{“}+\text{”}, \text{“}-\text{”}, \text{“}(\text{”}, \text{“})\text{”}\}$. The input terminates with a line containing a single "0" which should not be processed. To make your life easier, it is guaranteed that there is no space character in the input.

Output (Standard Output)

For each test case, output four lines on the standard output, where line i contains the percentage of votes given to the i^{th}choice. The percentages must be **truncated** to integers. Then, output the total number of participants in the poll, discarding duplicates SMSs. The format of the output must conform to the format indicated in the Standard Output below.

Sample Input and Output

Standard Input	Standard Output
6 09128122190:1 +98211100:+1 +98(912)812-2190:4 +311(20)590-4359:2 6616:1 02166-16-00-22:2 8 6616:1 0216616:2 +98216616:3 +98(21)66-16:4x 9090:1 8080:2: +1(519)708-2040:3 (519)708-2040:3:1 0	50% 50% 0% 0% Participants:5 66% 0% 33% 0% Participants:5