Training for String
Luke

Jan,20 2014

PROBLEM SET

This problem set should contain *9* problems on *22* numbered pages.

Description

A regular palindrome is a string of numbers or letters that is the same forward as backward. For example, the string "ABCDEDCBA" is a palindrome because it is the same when the string is read from left to right as when the string is read from right to left.

A mirrored string is a string for which when each of the elements of the string is changed to its reverse (if it has a reverse) and the string is read backwards the result is the same as the original string. For example, the string "3AIAE" is a mirrored string because "A" and "I"are their own reverses, and "3" and "E" are each others' reverses.

A mirrored palindrome is a string that meets the criteria of a regular palindrome and the criteria of a mirrored string. The string"ATOYOTA" is a mirrored palindrome because if the string is read backwards, the string is the same as the original and because if each of the characters is replaced by its reverse and the result is read backwards, the result is the same as the original string.

Of course,"A","T", "O", and "Y" are all their own reverses.

A list of all valid characters and their reverses is as follows.

Character	Reverse	Character	Reverse	Character	Reverse
A	A	M	M	Y	Y
B		N		Z	5
C		O	O	1	1
D		P		2	S
E	3	Q		3	E
F		R		4	
G		S	2	5	Z
H	H	T	T	6	
I	I	U	U	7	
J	L	V	V	8	8
K		W	W	9	

L	J	X	X		

Note that O (zero) and 0 (the letter) are considered the same character and therefore **ONLY** the letter "0" is a valid character.

Input

Input consists of strings (one per line) each of which will consist of one to twenty valid characters. There will be no invalid characters in any of the strings. Your program should read to the end of file.

Output

For each input string, you should print the string starting in column 1 immediately followed by exactly one of the following strings.

STRING	CRITERIA
″ -- is not a palindrome.″	if the string is not a palindrome and is not a mirrored string
″ -- is a regular palindrome.″	if the string is a palindrome and is not a mirrored string
″ -- is a mirrored string.″	if the string is not a palindrome and is a mirrored string
″ -- is a mirrored palindrome.″	if the string is a palindrome and is a mirrored string

Note that the output line is to include the -'s and spacing exactly as shown in the table above and demonstrated in the Sample Output below.

In addition, after each output line, you must print an empty line.

Sample Input

```
NOTAPALINDROME
ISAPALINILAPASI
2A3MEAS
ATOYOTA
```

Sample Output

```
NOTAPALINDROME -- is not a palindrome.

ISAPALINILAPASI -- is a regular palindrome.

2A3MEAS -- is a mirrored string.

ATOYOTA -- is a mirrored palindrome.
```

Hint

use the C++'s class of string

B - Where's Waldorf?

Description

Given a *m* by *n* grid of letters, ($1 \leq m, n \leq 50$), and a list of words, find the location in the grid at which the word can be found. A word matches a straight, uninterrupted line of letters in the grid. A word can match the letters in the grid regardless of case (i.e. upper and lower case letters are to be treated as the same). The matching can be done in any of the eight directions either horizontally, vertically or diagonally through the grid.

Input

The input begins with a single positive integer on a line by itself indicating the number of the cases following, each of them as described below. This line is followed by a blank line, and there is also a blank line between two consecutive inputs.

The input begins with a pair of integers, *m* followed by *n*, $1 \leq m, n \leq 50$ in decimal notation on a single line. The next *m* lines contain *n* letters each; this is the grid of letters in which the words of the list must be found. The letters in the grid may be in upper or lower case. Following the grid of letters, another integer *k* appears on a line by itself ($1 \leq k \leq 20$). The next *k* lines of input contain the list of words to search for, one word per line. These words may contain upper and lower case letters only (no spaces, hyphens or other non-alphabetic characters).

Output

The outputs of two consecutive cases will be separated by a blank line.

For each word in the word list, a pair of integers representing the location of the corresponding word in the grid must be output. The integers must be separated by a single space. The first integer is the line in the grid where the first letter of the given word can be found (1 represents the topmost line in the grid, and *m* represents the bottommost line). The second integer is the column in the grid where the first letter of the given word can be found (1 represents the leftmost column in the grid, and *n* represents the rightmost column in the grid). If a word can be found more than once in the grid, then the location which is output should correspond to the uppermost occurence of the word (i.e. the occurence which places the first letter of the word closest to the top of the grid). If two or more words are uppermost, the output should correspond to the leftmost of these occurences. All words can be found at least once in the grid.

Sample Input

```
1

8 11
abcDEFGhigg
hEbkWalDork
FtyAwaldORm
FtsimrLqsrc
byoArBeDeyv
Klcbqwikomk
strEBGadhrb
yUiqlxcnBjf
4
Waldorf
Bambi
Betty
Dagbert
```

Sample Output

```
2 5
2 3
1 2
7 8
```

Hint

Pay attention to the key phases in the problem.

C - Automatic Poetry

Description

"Oh God", Lara Croft exclaims, "it's one of these dumb riddles again!"

In Tomb Raider XIV, Lara is, as ever, gunning her way through ancient Egyptian pyramids, prehistoric caves and medival hallways. Now she is standing in front of some important Germanic looking doorway and has to solve a linguistic riddle to pass. As usual, the riddle is not very intellectually challenging.

This time, the riddle involves poems containing a "Schuttelreim". An example of a Schuttelreim is the following short poem:

Ein Kind halt seinen Schnabel nur,

wenn es hangt an der Nabelschnur.

/*German contestants please forgive me. I had to modify something as they were not appearing correctly in plain text format*/

A Schuttelreim seems to be a typical German invention. The funny thing about this strange type of poetry is that if somebody gives you the first line and the beginning of the second one, you can complete the poem yourself. Well, even a computer can do that, and your task is to write a program which completes them automatically. This will help Lara concentrate on the "action" part of Tomb Raider and not on the "intellectual" part.

Input

The input will begin with a line containing a single number n. After this line follow n pairs of lines containing Schuttelreims. The first line of each pair will be of the form

$$s_1<s_2>s_3<s_4>s_5$$

where the s_i are possibly empty, strings of lowercase characters or blanks. The second line will be a string of lowercase characters or blanks ending with three dots "...". Lines will we at most 100 characters long.

Output

For each pair of **Schuttelreim** lines $\mathbf{l_1}$ and $\mathbf{l_2}$ you are to output two lines $\mathbf{c_1}$ and $\mathbf{c_2}$ in the following way: $\mathbf{c_1}$ is the same as $\mathbf{l_1}$ only that the bracket marks "<" and ">" are removed. Line c_2 is the same as l_2 , except that instead of the three dots the string $\mathbf{s_4s_3s_2s_5}$ should appear.

Sample Input

```
3

ein kind haelt seinen <schn>abel<n>ur

wenn eshaengtan der ...

weil wirzuspaetzur<>oma<k>amen

verpassten wir das ...

<d>u <b>ist

...
```

Sample Output

```
ein kind haelt seinen schnabel nur

wenn es haengt an der nabel schnur

weil wir zu spaet zur oma kamen

verpassten wir das koma amen
```

du bist

bu dist

Hint

when u read a line of string after an int, u can use following codes, like

```
scanf("%d",&t);getchar();gets(str);
```

```
or    string str; cin>>t;getchar();getline(cin,str);
```

or u may read some unexpected words

D - Artificial Intelligence?

Description

Physics teachers in high school often think that problems given as text are more demanding than pure computations. After all, the pupils have to read and understand the problem first!

So they don't state a problem like ``U=10V, I=5A, P=?" but rather like ``You have an electrical circuit that contains a battery with a voltage of U=10V and a light-bulb. There's an electrical current of I=5A through the bulb. Which power is generated in the bulb?".

However, half of the pupils just don't pay attention to the text anyway. They just extract from the text what is given: U=10V, I=5A. Then they think: ``Which formulae do I know? Ah yes, P=U*I. Therefore P=10V*5A=500W. Finished."

OK, this doesn't always work, so these pupils are usually not the top scorers in physics tests. But at least this simple algorithm is usually good enough to pass the class. (Sad but true.)

Today we will check if a computer can pass a high school physics test. We will concentrate on the P-U-I type problems first. That means, problems in which two of power, voltage and current are given and the third is wanted.

Your job is to write a program that reads such a text problem and solves it according to the simple algorithm given above.

Input

The first line of the input file will contain the number of test cases.

Each test case will consist of one line containing exactly two data fields and some additional arbitrary words. A data field will be of the form I=*x*A, U=*x*V or P=*x*W, where *x* is a real number.

Directly before the unit (A, V or W) one of the prefixes m (milli), k (kilo) and M (Mega) may also occur. To summarize it: Data fields adhere to the following grammar:

```
        DataField ::= Concept '=' RealNumber [Prefix] Unit
Concept    ::= 'P' | 'U' | 'I'
Prefix     ::= 'm' | 'k' | 'M'
Unit       ::= 'W' | 'V' | 'A'
```

Additional assertions:

- The equal sign (`=') will never occur in an other context than within a data field.
- There is no whitespace (tabs,blanks) inside a data field.
- Either P and U, P and I, or U and I will be given.

Output

For each test case, print three lines:

- a line saying ``Problem #*k*'' where *k* is the number of the test case
- a line giving the solution (voltage, power or current, dependent on what was given), written without a prefix and with two decimal places as shown in the sample output
- a blank line

Sample Input

```
3
If the voltage is U=200V and the current is I=4.5A, which power is
generated?
A light-bulb yields P=100W and the voltage is U=220V. Compute the
current, please.
blablabla lightning strike I=2A blablabla P=2.5MW blabla voltage?
```

Sample Output

```
Problem #1
P=900.00W
```

```
Problem #2
I=0.45A

Problem #3
U=1250000.00V
```

Hint

double atof (const char* str);

Convert string to double

Parses the C string str, interpreting its content as a floating point

number and returns its value as a double.

----

when u read a line of string after an int, u can use following codes, like

```
scanf("%d",&t);getchar();gets(str);
```

or u may read some unexpected words

E - Excuses, Excuses!

Description

Judge Ito is having a problem with people subpoenaed for jury duty giving rather lame excuses in order to avoid serving. In order to reduce the amount of time required listening to goofy excuses, Judge Ito has asked that you write a program that will search for a list of keywords in a list of excuses identifying lame excuses. Keywords can be matched in an excuse regardless of case.

Input

Input to your program will consist of multiple sets of data.

- Line 1 of each set will contain exactly two integers. The first number ($1 \leq K \leq 20$) defines the number of keywords to be used in the search. The second number ($1 \leq E \leq 20$) defines the number of excuses in the set to be searched.
- Lines 2 through K+1 each contain exactly one keyword.
- Lines K+2 through K+1+E each contain exactly one excuse.
- All keywords in the keyword list will contain only contiguous lower case alphabetic characters of length L ($1 \leq L \leq 20$) and will occupy columns 1 through L in the input line.
- All excuses can contain any upper or lower case alphanumeric character, a space, or any of the following punctuation marks [SPMamp".,!?&] not including the square brackets and will not exceed 70 characters in length.
- Excuses will contain at least 1 non-space character.

Output

For each input set, you are to print the worst excuse(s) from the list.

- The worst excuse(s) is/are defined as the excuse(s) which contains the largest number of incidences of keywords.
- If a keyword occurs more than once in an excuse, each occurrance is considered a separate incidence.
- A keyword ``occurs" in an excuse if and only if it exists in the string in contiguous form and is delimited by the beginning or end of the line or any non-alphabetic character or a space.

For each set of input, you are to print a single line with the number of the set immediately after the string ``Excuse Set #". (See the Sample Output). The following line(s) is/are to contain the worst excuse(s) one per line exactly as read in. If there is more than one worst excuse, you may print them in any order.

After each set of output, you should print a blank line.

Sample Input

```
5 3
dog
ate
homework
canary
died
My dog ate my homework.
Can you believe my dog died after eating my canary...AND MY HOMEWORK?
This excuse is so good that it contain 0 keywords.
6 5
superhighway
crazy
thermonuclear
bedroom
war
building
I am having a superhighway built in my bedroom.
I am actually crazy.
1234567890.....,,,,,0987654321?????!!!!!!
There was a thermonuclear war!
I ate my dog, my canary, and my homework ... note outdated keywords?
```

Sample Output

```
Excuse Set #1
Can you believe my dog died after eating my canary...AND MY HOMEWORK?

Excuse Set #2
I am having a superhighway built in my bedroom.
There was a thermonuclear war!
```

F - Decode the tape

Description

"Machines take me by surprise with great frequency."

Alan Turing

Your boss has just unearthed a roll of old computer tapes. The tapes have holes in them and might contain some sort of useful information. It falls to you to figure out what is written on them.

Input

The input will contain one tape.

Output

Output the message that is written on the tape.

Sample Input

```
___________
| o   .  o|
|  o  .   |
| ooo .  o|
| ooo .o o|
| oo o.  o|
| oo  .oo|
| oo o. oo|
|  o  .   |
| oo  .o |
| ooo .o |
| ooo.ooo|
| ooo .ooo|
| ooo.oo |
|  o  .   |
| oo  .oo |
| ooo.ooo|
| oooo.   |
|  o  .   |
| oo o. o |
| ooo .o o|
| ooo.o o|
| ooo .   |
| ooo .oo|
|  o  .   |
| ooo.ooo|
| ooo .oo |
| oo  .o o|
| ooo .o |
|  o  .   |
| ooo .o  |
| oo o.   |
| oo  .o o|
|  o  .   |
| ooo.o  |
| oo  .  o|
| oooo. o |
| oooo.  o|
|  o  .   |
| oo  .o  |
| ooo.ooo|
| oo  .ooo|
|  oo.oo |
|    o. o |
```

Sample Output

```
A quick brown fox jumps over the lazy dog.
```

G - Andy's Second Dictionary

Description

Andy is now 9-year old and is getting ambitious. He wants to become the world largest dictionary editor. You have already helped him to solve a problem with a computer and he now turns back to you with a new challenge. He already has a program that copies all the words from a text and outputs them **in alphabetical order**. Unfortunately, this program does not take into account hyphenation and is not satisfying him. He wants you to write a new program that can copy words from a text, even when those words are hyphenated.

The input file is a text with no more than 500 words of arbitrary length. Input is terminated by EOF.

You are asked to write a program that lists all the different words in the input text. In this problem, a word is defined as a consecutive sequence of alphabets, in upper and/or lower case. Words with only one letter are also to be considered. Furthermore, your program must be CaSe InSeNsItIvE. For example, words like "Apple", "apple" or "APPLE" must be considered the same. **Also words may be hyphenated.** When a word is hyphenated, its prefix initiates on a line, followed by hyphen character, maybe followed by a newline, followed by the rest of the word. The rest of the word may be itself hyphenated. The hyphen character is part of the word if, and only if, not followed by a newline.

Input

The input file is a text terminated by EOF.

Output

Your output should give a list of different words that appears in the input text, one in a line. The words should all be in lower case, sorted in alphabetical order.

Sample Input

```
Adv-
ent-
ures
in
Dis-
ney-
land

Two blondes were go-
ing  to  Disney-land
when they  came to a
fork  in  the  road.
The sign read: "Dis-
neyland Left."

So they went home.
```

Sample Output

```
a
adventures
blondes
came
disney-land
disneyland
fork
going
home
in
left
read
road
sign
so
the
they
to
two
went
were
when
```

H - Immediate Decodability

Description

An encoding of a set of symbols is said to be *immediately* decodable if no code for one symbol is the prefix of a code for another symbol. We will assume for this problem that all codes are in binary, that no two codes within a set of codes are the same, that each code has at least one bit and no more than ten bits, and that each set has at least two codes and no more than eight.

Examples: Assume an alphabet that has symbols `{A, B, C, D}`

The following code is immediately decodable:

```
A:01 B:10 C:0010 D:0000
```

but this one is not:

`A:01 B:10 C:010 D:0000` (Note that `A` is a prefix of `C`)

Input

Write a program that accepts as input a series of groups of records from a data file. Each record in a group contains a collection of zeroes and ones representing a binary code for a different symbol. Each group is followed by a single separator record containing a single 9; the separator records are not part of the group. Each group is independent of other groups; the codes in one group are not related to codes in any other group (that is, each group is to be processed independently).

Output

For each group, your program should determine whether the codes in that group are immediately decodable, and should print a single output line giving the group number and stating whether the group is, or is not, immediately decodable.

Sample Input

```
01
10
0010
0000
9
01
10
010
0000
9
```

Sample Output

```
Set 1 is immediately decodable
Set 2 is not immediately decodable
```

Hint

int strncmp (const char * str1, const char * str2, size_t num);

Compare characters of two strings

Compares up to num characters of the C string str1 to those of the C string str2.

I - Automatic Editing

Description

Text-processing tools like *awk* and *sed* allow you to automatically perform a sequence of editing operations based on a s cript. For this problem we consider the specific case in which we want to perform a series of string replacements, within a single line of text, based on a fixed set of rules. Each rule specifies the string to find, and the string to replace it with, as shown below.

Rule	Find	Replace-by
1.	ban	bab
2.	baba	be
3.	ana	any
4.	ba b	hind the g

To perform the edits for a given line of text, start with the first rule. Replace the first occurrence of the *find* string within the text by the*replace-by* string, then try to perform the same replacement *again* on the new text. Continue until the *find* string no longer occurs within the text, and then move on to the next rule. Continue until all the rules have been considered. Note that (1) when searching for a *find*string, you always start searching at the beginning of the text, (2) once you have finished using a rule (because the *find* string no longer occurs) you never use that rule again, and (3) case is significant.

For example, suppose we start with the line

```
banana boat
```

and apply these rules. The sequence of transformations is shown below, where occurrences of a *find* string are underlined and replacements are boldfaced. Note that rule 1 was used twice, then rule 2 was used once, then rule 3 was used zero times, and then rule 4 was used once.

Before	After
banana boat	**bab**ana boat
babana boat	ba**bab**a boat
bababa boat	**be**ba boat
beba boat	be**hind the g**oat

Input

The input contains one or more test cases, followed by a line containing only 0 (zero) that signals the end of the file. Each test case begins with a line containing the number of rules, which will be between 1 and 10. Each rule is specified by a pair of lines, where the first line is the *find* string and the second line is the *replace-by* string. Following all the rules is a line containing the text to edit.

Output

For each test case, output a line containing the final edited text.

Both *find* and *replace-by* strings will be at most 80 characters long. *Find* strings will contain at least one character, but *replace-by* strings may be empty (indicated in the input file by an empty line). During the edit process the text may grow as large as 255 characters, but the final output text will be less than 80 characters long.

Sample Input

```
4
ban
bab
baba
be
ana
any
ba b
hind the g
banana boat
1
t
sh
toe or top
```

```
0
```

Sample Output

```
behind the goat
shoe or shop
```

Hint

when u read a line of string after an int, u can use following codes, like

```
scanf("%d",&t);getchar();gets(str);
```

or u may read some unexpected words