Labatt Contest Problems

Problem 1 – Kayaks

Suppose we have an even number of people going on an excursion, and we would like to fit them into two-seated kayaks. All kayaks are identical, weighing 20kg each. Each person is described by two parameters, their *strength* and *weight*. The speed of a kayak can be calculated as the sum of strengths of both persons sitting in it, divided by the total weight of the loaded kayak (i.e., the weight of the kayak plus the weight of both persons). We would like to choose the allocation of people to kayaks so as to maximize the speed at which the whole whole group can travel, assuming that the group travels at the speed of the slowest kayak in it.

Input

First, $2 \le n \le 10^5$, the number of people (n will always be even). Then, n pairs of integers follow, each pair describing one person: first, $20 \le w \le 300$, the weight of the person (in kilograms), then $20 \le p \le 300$, the strength of the person.

Output

Output one number: the maximum speed of the group which can be achieved by optimally choosing places for each person. The answer should be accurate up to 6 digits after the decimal point.

Example

Input:

```
4
50 50
50 60
70 100
100 60
```

Output:

```
0.842105
```

Problem 2 – Poker

In poker, you have 5 cards. There are 10 kinds of poker hands (from highest to lowest):

- royal flush - ace, king, queen, jack and ten, all in the same suit
- straight flush - five cards of the same suit in sequence, such as 10,9,8,7,6 of clubs; ace can be counted both as the highest card or as the lowest card - A,2,3,4,5 of hearts is a straight flush. But 4,3,2,A,K of hearts is not a straight flush - it's just a flush.
- four of a kind - four cards of the same rank, such as four kings.
- full house - three cards of one rank plus two cards of another rank
- flush - five cards of the same suit (but not a straight flush)
- straight - five cards in order - just like the straight flush, but mixed suits
- three of a kind - three cards of one rank and two other cards
- two pairs - two cards of one rank, two cards of another rank, and one more card
- pair - two cards of the same rank
- high card - none of the above

Write a program that will help you play poker by telling you what kind of hand you have.

Input

The first line of input contains the number of test cases (no more than 20). Each test case consists of one line - five space separated cards. Each card is represented by a two-letter (or digit) word. The first character is the rank (A,K,Q,J,T,9,8,7,6,5,4,3 or 2), the second character is the suit (S,H,D,C standing for spades, hearts, diamonds and clubs). The cards can be in any order (but they will not repeat).

Output

For each test case output one line describing the type of a hand, exactly like in the list above.

Example

Input:
```
3
AH KH QH TH JH
KH 5S 3C 5C 7D
QH QD 2S QC 2C
```

Output:
```
royal flush
pair
full house
```

Problem 3 – Artificial Intelligence?

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 as ``U=10V, I=5A, P=?" but rather as ``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. How much 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 formula 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 another 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, how much power is
generated?
A light-bulb yields P=100W and the voltage is U=220V. Compute the current,
please.
bla bla bla lightning strike I=2A bla bla bla P=2.5MW bla bla voltage?
```

Sample Output

```
Problem #1
P=900.00W

Problem #2
I=0.45A

Problem #3
U=1250000.00V
```

Problem 4 – Hay Points

Each employee of a bureaucracy has a *job description* - a few paragraphs that describe the responsibilities of the job. The employee's job description, combined with other factors, such as seniority, is used to determine his or her salary.

The *Hay Point* system frees the Human Resources department from having to make an intelligent judgment as to the value of the employee; the job description is merely scanned for words and phrases that indicate responsibility. In particular, job descriptions that indicate control over a large budget or management over a large number of people yield high Hay Point scores.

You are to implement a simplified Hay Point system. You will be given a Hay Point dictionary and a number of job descriptions. For each job description you are to compute the salary associated with the job, according to the system.

The first line of input contains 2 positive integers: *m <= 1000*, the number of words in the Hay Point dictionary, and *n <= 100*, the number of job descriptions. *m* lines follow; each contains a word (a string of up to 16 lower-case letters) and a dollar value (a real number between 0 and 1,000,000). Following the dictionary are the *n* job descriptions. Each job description consists of one or more lines of text; for your convenience the text has been converted to lower case and has no characters other than letters, numbers, and spaces. Each job description is terminated by a line containing a period.

For each job description, output the corresponding salary computed as the sum of the Hay Point values for all words that appear in the description. Words that do not appear in the dictionary have a value of 0.

Sample Input

```
7 2
administer 100000
spending 200000
manage 50000
responsibility 25000
expertise 100
skill 50
money 75000
the incumbent will administer the spending of kindergarten milk money
and exercise responsibility for making change he or she will share
responsibility for the task of managing the money with the assistant
whose skill and expertise shall ensure the successful spending exercise
.
this individual must have the skill to perform a heart transplant and
expertise in rocket science
.
```

Output for Sample Input

```
700150
150
```

Problem 5 – Maze Traversal

A common problem in artificial intelligence is negotiation of a maze. A maze has corridors and walls. The robot can proceed along corridors, but cannot go through walls.

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.

For this problem, you will input the dimensions of a maze, as two integer values on the first line. Of the two numbers, the first is the number of rows and the second is the number of columns. Neither the number of rows nor columns will exceed 60.

Following these numbers will be a number of rows, as specified previously. In each row there will be a character for each column, with the row terminated by the end of line. Blank spaces are corridors, asterisks are walls. There needn't be any exits from the maze.

Following the maze, will be an initial row and column specified as two integers on one line. This gives the initial position of the robot. Initially the robot will be facing North (toward the first row).

The remaining input will consist of commands to the robot, with any amount of interspersed white-space. The valid commands are:

R

rotate the robot 90 degrees clockwise (to the right)

L

rotate the robot 90 degrees counter-clockwise (to the left)

F

move the robot forward, unless a wall prevents this, in which case do nothing

Q

quit the program, printing out the current robot row, column and orientation

Output

For each test case, the output must follow the description below. The outputs of two consecutive cases will be separated by a blank line.

The final row and column must be integers separated by a space. The orientation must be one of N,W,S,E and separated from the column by a space.

Sample Input

```
1

7 8
********
* * * **
* *    *
* * ** *
* * *  *
*   * **
********
2 4
RRFLFF FFR
FF
RFFQ
```

Sample Output

```
5 6 W
```

Problem 6 – Roman Numeral

Context

This year is the XXV Anniversary of the Faculty of Computer Science in Murcia. But, what does XXV means? Maybe you should ask an ancient Roman to get the answer.

The Problem

A Roman numeral consists of a set of letters of the alphabet. Each letter has a particular value, as shown in the following table:

Letter	I	V	X	L	C	D	M
Value	1	5	10	50	100	500	1000

Generally, Roman numerals are written in descending order from left to right, and are added sequentially. However, certain combinations employ a subtractive principle. If a symbol of smaller value precedes a symbol of larger value, the smaller value is subtracted from the larger value, and the result is added to the total. This subtractive principle follows the next rules:

- "I" may only precede "V" and "X" (e.g., IV=4).
- "X" may only precede "L" and "C" (e.g., XC=90).
- "C" may only precede "D" and "M".
- "V", "L" and "D" are always followed by a symbol of smaller value, so they are always added to the total.

Symbols "I", "X", "C" and "M" cannot appear more than three consecutive times. Symbols "V", "L" and "D" cannot appear more than once consecutively.
Roman numerals do not include the number zero, and for values greater or equal than 4000 they used bars placed above the letters to indicate multiplication by 1000.
You have to write a program that converts from Roman to Arabic numerals and vice versa.
Although lower case letters were used in the Middle Ages, the Romans only used upper case letters. Therefore, for the Roman numerals we only consider upper case letters.

The Input

The input consists of several lines, each one containing either an Arabic or a Roman number *n*, where $0 < n < 4000$.

The Output

For each input line, you must print a line with the converted number. If the number is Arabic, you must give it in Roman format. If the number is Roman, you must give it in Arabic format.

Sample Input

```
XXV
4
942
MCMLXXXIII
```

Sample Output

```
25
IV
CMXLII
1983
```