2019 Consortium for Computing Sciences in Colleges
Programming Contest
Saturday, October 26th
Auburn University
Auburn, AL

There are nine (9) problems in this packet. Each team member should have a copy of the problems. These problems are NOT necessarily sorted by difficulty. You may solve them in any order.

Remember input/output for the contest will be from `stdin` to `stdout`. `Stderr` will be ignored. Do not refer to or use external files in your source code. Extra white space at the end of lines is ignored, but extra white space at the beginning or within text on a line is not ignored. Have a lot of fun and good luck! ☺

Problem 1. Multiple-Choice Exam

Problem 2. Health App

Problem 3. Transpose a Matrix

Problem 4. Slicing Potatoes

Problem 5. Fly-by Photos

Problem 6. Dismal Deals

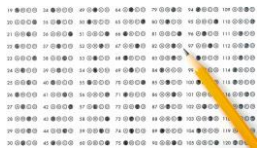
Problem 7. Kanye

Problem 8. MP3 Library

Problem 9. Day of the Year

Problem 1

Multiple-Choice Exam



You are writing a computer program to grade the responses on a recent 25 question multiple-choice exam. Each question has four options only (A, B, C, or D) and is worth four points each. Your program should calculate the high score on the exam, the class median, and report the question number answered most correctly and incorrectly. The median is the middle number in a given sequence of numbers, taken as the average of the two middle numbers when the sequence is an even number.

Input

The first line of input is a string of length 25 representing the correct answers on the key. A line starting with a positive integer *n* less than 100 representing the number of exams to be graded follows on the second line. This is followed by *n* lines each containing a string of length 25 representing the particular responses given by a test taker.

Output

Output your four-line summary in the exact format as seen below. The high score and median should not be rounded and will range from a low of 0 up to a high of 100. Always use exactly one decimal point of precision on the median value. Put a `%` character after both the high score and median as well as a space on both sides of all equals sign. In the event of a tie on the question answered most correctly or incorrectly, use the lowest question number. The first question is numbered 1 and the last question is numbered 25.

Sample Input

```
CDBBACAABBABCCDCDADCBBAAB
12
CDBBDCAABBABCCDCBADCBDACB
CDBBBCAABBABCCDCDADCBBADB
BDBBACBABBABCCDCDADCBCAAB
DDBBACAABBABCADCDACCBBAAB
ADBBCDAADDABBBDCDADCBBADB
CDBBACABABABCCDCDADCBBAAB
ABABABABABABABABABABABABA
DDBBACAABBABCCDCDADCBBAAB
CDBBACAABBABCCDCDADCBBDDB
CDBBACAACBABBCDCDADCBBBBB
CCCCCCCCCCCCCCCCCCCCCCCCC
BBBBBBBBBBBBBBBBBBBBBBBBB
```

Output Corresponding to Sample Input

```
High Score = 96%
Median Score = 86.0%
Question Number Answered Most Correctly = 4
Question Number Answered Most Incorrectly = 24
```

Problem 2

Health App

Will is upgrading to a new iphoneX and wants to download all his walking data from the health app of his old iphone. This health app acts as a pedometer and records his daily steps and mileage. After downloading, he would like to summarize the weekly, monthly, and year to date mileage before importing to his new phone. Your job is to write a program which will give him a summary of his weekly, monthly, and yearly data.

Input

The input to your program will contain at least one week of data. All inputs are floating point values representing a distance in miles except for negative inputs which are used as delimiters. A `-1.0` means end of week, a `-2.0` means end of month, and a `-3.0` means end of input and the year. A month will always follow at least four weeks of data, and may end at any point during a week. A week may have fewer than seven readings if it falls at the beginning or ending of the year or at the end of the input. A `-3.0` may occur at any time following a `-1.0`.

Output

Output your summary in the exact format as seen below with counters for each week and month. Each average mileage should be rounded to the nearest hundredth and displayed to exactly two decimal places. A blank line should be printed before and after each month and year summary. The year summary should include the total days recorded by the pedometer.

Sample Input

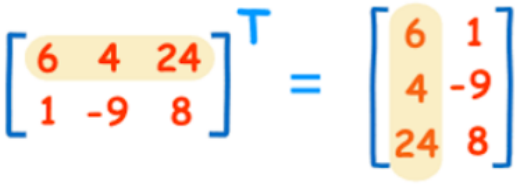
```
7.2 4.0 2.7 4.6 5.7 -1.0
4.9 5.1 4.9 5.6 4.9 0.3 1.1 -1.0
7.1 2.9 1.8 0.9 5.9 5.8 4.9 -1.0
4.4 4.8 5.3 6.2 4.2 1.2 1.7 -1.0
2.3 4.3 1.9 0.4 3.5 -2.0 5.2 1.1 -1.0
1.6 2.2 4.2 5.0 1.7 5.9 0.9 -1.0
3.3 2.0 4.9 5.1 4.9 5.6 4.9 -1.0
1.1 2.9 1.8 0.9 5.9 5.8 4.9 -1.0
9.8 3.4 4.8 5.3 6.2 -2.0 4.7 5.1 -1.0
1.3 3.0 3.1 3.1 4.9 5.6 4.6 -1.0
4.1 3.0 -1.0 -3.0
```

Output Corresponding to Sample Input

```
Week #1 = 4.84 mi.
Week #2 = 3.83 mi.
Week #3 = 4.19 mi.
Week #4 = 3.97 mi.

Month #1 = 3.89 mi.

Week #5 = 2.67 mi.
Week #6 = 3.07 mi.
Week #7 = 4.39 mi.
Week #8 = 3.33 mi.

Month #2 = 3.98 mi.

Week #9 = 5.61 mi.
Week #10 = 3.66 mi.
Week #11 = 3.55 mi.

Year to Date for 70 days = 3.92 mi.
```

Problem 3

Transpose a Matrix



The transpose of a matrix is a new matrix whose rows are the columns of the original. Above is a matrix and its transpose. The superscript *T* means "transpose". Another way to look at the transpose is that the element at row *r* column *c* in the original is placed at row *c* column *r* of the transpose. Your job is to write a program which will produce a transpose of a matrix.

Input

The first line of input will contain a single integer *n*, which represents the number of test cases. You may assume that $1 \leq n \leq 100$. This will be followed by *n* test cases. Each test case consists of a single line with a positive integer *r*, $1 \leq r \leq 100$ representing the number of rows, a single space, and a positive integer *c*, $1 \leq c \leq 100$ representing the number of columns. This is followed by *r* rows and *c* columns of positive integers in the matrix to be transposed. Each integer is separated by a single space. Your input will be in row major order.

Output

For each matrix input, you should print out the transpose matrix in the format as seen below. Include a blank line between each test case.

Sample Input

```
3
2 3
6 4 24
1 -9 8
3 3
1 2 3
4 5 6
7 8 9
2 1
5
4
```

Output Corresponding to Sample Input

```
T of Matrix 1:
6 1
4 -9
24 8

T of Matrix 2:
1 4 7
2 5 8
3 6 9

T of Matrix 3:
5 4
```

Problem 4

Slicing Potatoes

Vera is cooking dinner. Her little sister Alice wants to help. So, Vera asks Alice to slice the potatoes that she is cooking. Vera instructs Alice to make sure that when slicing a potato, all of the slices have the same thickness. Alice asks why, but Vera does not answer. So, Alice begins to wonder how big each of the slices would be when she cuts each potato into equal-width slices. Let us write a program to help Alice calculate the volume of each potato slice. For the purpose of this problem, let us assume that a potato is geometrically represented as an ellipsoid having this Cartesian equation:

$$\frac{x^2}{L^2} + \frac{y^2 + z^2}{W^2} = 1$$

where the length of the potato is *2L* and the width is *2W*. You may assume that $L > W$. In other words, this solid could be formed by revolving the ellipse

$$\frac{x^2}{L^2} + \frac{y^2}{W^2} = 1$$

about the x-axis.

When a potato is sliced, the cuts are made along planes that are parallel to the *yz* plane. If we want to cut a potato into n slices, then we need to make $n - 1$ cuts. The cuts are made so that each slice of the potato has the same thickness. In addition, since the cuts are made across the whole length of the potato, the thickness of each slice must be as follows.

$$\frac{2L}{n}$$

An instance of this problem consists of three numbers:

- A real number representing the length of the potato
- A real number representing the width of the potato
- An integer representing the desired number of slices for this potato. You may assume that the number of slices will be at least two, but less than 100.

Your program will compute the volume of each slice, in order from one end of the potato to the other.

Input
The first line of the input will contain *P*, an integer representing the number of potatoes. You may assume that *P* is a positive integer less than 100. Each of the next *P* lines of input contains three numbers, representing the information about one potato: the length, the width, and number of slices. Assume one or more space characters separate the three numbers.

Output
For each potato, your program needs to print a line that says `Potato <n>`, where n is the potato number and $1 \leq n \leq$ P. Underneath the potato number, the volume for each slice should be printed on its own line.

Print information about a slice as follows:

`Slice <n> = <volume>`, where `n` is the slice number starting with 1, and the volume is specified to exactly three decimal places of precision.

Each line of slice information needs to be indented by two spaces. In addition, there must be a single space on both sides of the equals sign.

Sample Input
```
2
4.0 2.0 3
5.25 3.5 4
```

Output Corresponding to Sample Input
```
Potato 1
  Slice 1 = 2.172
  Slice 2 = 4.034
  Slice 3 = 2.172
Potato 2
  Slice 1 = 5.262
  Slice 2 = 11.575
  Slice 3 = 11.575
  Slice 4 = 5.262
```

Problem 5

Fly-by Photos

NASA is preparing for an upcoming mission where a satellite will be taking fly-by photos of planets in our solar system. However, before launching the probe, they have asked you to simulate a few scenarios so they can estimate how good of a picture they could take.

They have described the situation as follows. The satellite follows a linear path, and can be modeled like this:

$$Sx(t) = x_0 + v_x * t$$
$$Sy(t) = y_0 + v_y * t$$

where x_0 and y_0 are the initial position of the satellite, and v_x and v_y are the velocity of the satellite in the x and y components respectively.

The planets they are modeling follow an elliptical path. That path is modeled like this:

$$Ex(t) = A * cos(f * t)$$
$$Ey(t) = B * sin(f * t)$$

where A is the orbital radius in the *x*-dimension, B is the orbital radius in the *y*-dimension, and f is how fast the planet orbits the sun.

Given these formulae for the two objects, NASA has requested that you find the point at which the satellite and planet are closest so that they can take the best picture possible.

To simplify computations, NASA has guaranteed that each scenario given will only last for 10 seconds. After that, you may assume that the satellite has already passed its target.

NASA would also like to remind you that to find the distance between the two targets for some value of t, you will need to calculate:

$$D(t) = \sqrt{(Sx(t) - Ex(t))^2 + (Sy(t) - Ey(t))^2}$$

Input
The first line of input is a positive integer, less than 100 denoting the number of scenarios to follow. The next line will consist of the parameters for each of the equations in the following order: A, B, f, x_0, y_0, v_x, and finally v_y. All of these numbers will be formatted to three decimal places. They can be negative, and will always lie in the range from a low of `-50.000` up to a high of `50.000`.

Output
Your output you should print the closest distance achieved between the satellite and the planet between $t = 0$ and $t = 10$, formatted to five decimal places.

Sample Input
```
2
2.000 1.000 2.000 2.000 -2.000 0.000 0.637
2.000 1.000 2.000 3.000 -2.000 0.000 1.000
```

Output Corresponding to Sample Input
```
0.00000
1.47178
```

Problem 6

Dismal Deals

Dismaland Autos has a unique business strategy. Every car listing on their website is displayed as a pair *dismal operations*; that is, each listing is an operation displayed assuming the use of dismal arithmetic, which is a simpler form of arithmetic developed and discussed by Neil Sloane and colleagues as a response to the "dismal state of arithmetic skills possessed by today's children." Dismal arithmetic only has two possible operations:

> "There are no carries, when you add digits you just take the largest, and when you multiply digits you take the smallest."

```
  11        123
+  6      x 123
----      -----
  16        123
           122
          111
          -----
          11223
```

Dismaland only displays listings briefly though, and purchases must be locked in by selecting one of the operations before the listing is removed. To avoid getting a dismal deal on the car you want, you need to program a calculator to evaluate each listing and determine which operation to pick. Given a listing with two expressions, your calculator must evaluate them and output the results with a <, >, or = symbol denoting their relationship.

Input

The first line of input contains a single integer n, ($1 \leq n \leq 10^6$), which is the number of test cases to follow. The next n lines will each contain two pairs of integers ($0 \leq x \leq 10^{10}$-1) separated by a space. Each pair of integers is separated with a dismal operator, either + or *, indicating the operation to be performed.

Output

For your output, you should print the comparison of the two operations. The result on the left will either be <, >, or = to the result on the right. Your output should be the first result, the correct comparison symbol, and the second result with a single space between each of these three items.

Sample Input

```
3
12*10 11*10
314*5 301+15
11*11 11+6
```

Output Corresponding to Sample Input

```
110 = 110
314 < 315
111 > 16
```

Problem 7

Kanye

Kanye has finally decided what he really loves most in this world... himself. He loves reading, but what he loves most is... himself. Some texts just really are not about Kanye that much, so he has asked you to fix that for him. Basically, he needs you to replace pronouns with "Kanye", so that the text talks more about Kanye.

However, the thing that Kanye loves second most is... grammar. So, he has asked you to ignore the pronoun "I" since replacing that might mess up subject verb agreement, which Kanye does not like. To help you out, Kanye has provided a table of how to replace pronouns.

Original	Replaced
me	Kanye
you	Kanye
she	Kanye
her	Kanye
he	Kanye
him	Kanye
my	Kanye's
mine	Kanye's
your	Kanye's
yours	Kanye's
hers	Kanye's
his	Kanye's

Input

Input will start with a single number telling how many lines will be in the text. The next n lines will contain text for you to "fix". Each line will be at most 80 characters in length. Note that case does not matter, and there can be punctuation in the string. Also, note that a word is defined as a sequence of alphanumeric characters, which means that punctuation defines a word boundary.

Output

For your output, you should simply print out the provided text, with the specified replacements applied.

Sample Input

```
2
He gave that to her
is that your cup or mine
```

Output Corresponding to Sample Input

```
Kanye gave that to Kanye
is that Kanye's cup or Kanye's
```

Problem 8

MP3 Library

You are parsing through a massive music library of MP3 files from multiple albums of your favorite artist. You want to develop a program which will quickly summarize the total songs and time you possess of that artist as well as a summary by album.

Input

The input is in a very specific format consisting of two or more songs. Each line contains exactly five words that describe a song of your favorite artist in your MP3 library as follows.

```
Title   Time (m:ss)   Album   Genre   Track
```

None of the words have spaces -- where there should be a space, there is instead an underscore. Track is a primary key, and you may assume no album has songs with the same track number. The last line of input will be a line with string `-1`, and should not be processed.

Output

Your program should first print out the total MP3 files in your collection followed by the total time as seen below. This should be followed by a blank line. This is followed by the name of the album, a colon, a space, the number of songs, a comma, a space, and total time for that album in the format `m:ss`. Album names should be appear sorted by title in ascending order.

After each album, you will print out the title of each song on that album, sorted by track number. The format of each of these lines will be a single space, the track number, a period, a space, the song's name, a colon, a space and the song's time. All underscores should be turned back into spaces. There should be no blank lines in the output after the one on the second line.

Sample Input

```
Down_In_Brazil 6:07 Naima Jazz 4
Naima 7:49 Naima Jazz 6
Naima 8:38 Eastern_Rebellion Jazz 2
This_Guy's_In_Love_With_You 8:10 Naima Jazz 2
-1
```

Output Corresponding to Sample Input

```
4 MP3 files, 30:44 of time

Eastern Rebellion: 1, 8:38
 2.  Naima 8:38
Naima: 3, 22:06
 2.  This Guy's In Love With You 8:10
 4.  Down In Brazil 6:07
 6.  Naima 7:49
```

Problem 9

Day of the Year

We want to write a program which takes a date in a format like 8/26/19 and tells us which numerical day of the year out of 365 it represents. To help us with this, use this classic song which helps you remember the number of days in a month.

30 days has September,
April, June and November
All the rest have 31
And February's great with 28
And Leap Year's February's fine with 29

Remember that a leap year is defined as follows.

> Every year that is exactly divisible by four is a leap year, except for years that are exactly divisible by 100, but these centurial years are leap years if they are exactly divisible by 400. For example, the year 1700 is not a leap year, but the years 1600 and 2000 are.

In addition to calculating the day of the year a date represents, we also want to know which day of the week it represents. Zeller's congruence is an algorithm devised by Christian Zeller to help us calculate the day of the week for any Gregorian calendar date. Zeller's congruence relies on the following quantities: `J` is the century (`19`, for example if the year input is `1998`), `K` is the year within the century (`98`, for example if the year input is `1998`), `m` is the month (where March is 3, April is 4, etc.), `q` is the day of the month. The day of the week is determined by this formula:

```
h = (q+26(m+1)/10+K+K/4+J/4+5J) mod 7
```

where the results of all divisions are truncated. The value of `h` will lie between 0 (Saturday) and 6 (Friday). Note that the expression a `mod b` yields the remainder produced by `a ÷ b`. Zeller's congruence assumes that January and February are treated as months `13` and `14` of the previous year; this affects the values `K` and `m`, and possibly the value of `J`. That is, January will never be a 1 and February a 2 (rather they are special cases where January will be 13 of the year prior and February 14 of the year prior).

Input
The first line of input contains a single integer n, ($1 \leq n \leq 1000$), which is the number of test cases that follow. Each test case consists of a single line of input containing a string. Each string is of length n, ($1 \leq n \leq 80$). The string will have the form `month/date/year` representing a valid date where month is in the range [1,12], date is in the range [1,31], and year is in the range [1600, 2100].

Output

For each test case, generate one line of output showing the date input followed by its day of week and its numerical day of the year in the exact format shown below. If the day of the year ends in a 1, follow it with the suffix `st`. If it ends with a 2, follow it with the suffix `nd`. If it ends with a 3, follow it with the suffix `rd`. For all others, follow it with the suffix `th`.

Sample Input
```
5
6/12/2019
1/2/2020
3/1/2020
3/1/2019
12/31/2020
```

Output Corresponding to Sample Input
```
6/12/2019 is a Wednesday and the 163rd day of the year.
1/2/2020 is a Thursday and the 2nd day of the year.
3/1/2020 is a Sunday and the 61st day of the year.
3/1/2019 is a Friday and the 60th day of the year.
12/31/2020 is a Thursday and the 366th day of the year.
```