University of Scranton
ACM Student Chapter / Computing Sciences Department
25th Annual High School Programming Contest (2015)

Problem 1: Days, Hours, and Minutes

Develop a program that, given a duration expressed solely in terms of minutes (we call this an **M-duration**), translates it into a duration expressed in terms of days, hours, and minutes (a **DHM-duration**).

For example, an M-duration of 7400 minutes translates into a DHM-duration of five days, three hours, and 20 minutes. Note that to be in proper form, the DHM-duration X days, Y hours, and Z minutes must satisfy the conditions $0 \leq X$, $0 \leq Y < 24$, and $0 \leq Z < 60$.

Input: The first line contains a positive integer n indicating how many M-durations are to be translated. Each of the following n lines contains a nonnegative integer expressing an M-duration.

Output: For each M-duration given as input, one line of output should be produced. That line should indicate the given M-duration and the equivalent DHM-duration in the format exemplified by the sample output below.

Duration Facts: There are 60 minutes in an hour and 24 hours in a day.

```
Sample input:
-------------
5
155
7914
8718
180
90725

Resultant output:
-----------------
155 Minutes is 0 Days 2 Hours 35 Minutes
7914 Minutes is 5 Days 11 Hours 54 Minutes
8718 Minutes is 6 Days 1 Hours 18 Minutes
180 Minutes is 0 Days 3 Hours 0 Minutes
90725 Minutes is 63 Days 0 Hours 5 Minutes
```

Problem 2: Calendar Date Conversion

Calendar dates are often expressed in the form `MM/DD/YYYY`, as in `03/18/1985`, which refers to the 18th day of the 3rd month of the year 1985.

Develop a program that, given a calendar date expressed in the `MM/DD/YYYY` format, translates it into the traditional format, which is `<month-name> <day>, <year>`. For example, `03/18/1985` translates to `March 18, 1985`. If the date given is not valid, identify it as such.

Input: The first line contains a positive integer n indicating how many calendar dates, each in the `MM/DD/YYYY` format, are to be found on the succeeding n lines, one per line.

You may assume that each `M`, `D`, and `Y` is a digit in the range `0..9`. However, a given date may be invalid by indicating a month that is outside the range 1..12 or by indicating a day that is outside the range 1..d, where d is the number of days in the specified month-year.

Output: For each `MM/DD/YYYY`-formatted calendar date given as input, display it, followed by a colon and a space, followed either by the date expressed in the traditional format or by the message `Invalid`, as appropriate. No leading zeros are to appear in numerals.

Calendar Facts: The months, in order from 1st to 12th, are `January`, `February`, `March`, `April`, `May`, `June`, `July`, `August`, `September`, `October`, `November`, and `December`.

Each of April, June, September, and November has 30 days, while each of January, March, May, July, August, October, and December has 31. February has 28 days, except during leap years, when it has 29. A leap year is one that is divisible by 400 **or** both divisible by four but not by 100. For example, 1900 was not a leap year, 2000 was, but 2100 will not be.

```
Sample input:
-------------
5
12/25/1976
02/29/2001
09/15/1956
05/47/1845
07/02/1999


Resultant output:
-----------------
12/25/1976: December 25, 1976
02/29/2001: Invalid
09/15/1956: September 15, 1956
05/47/1845: Invalid
07/02/1999: July 2, 1999
```

Problem 3: Ambiguous Calendar Dates

Calendar dates are often expressed in the form `MM/DD/YYYY`, as in `05/12/85`, which refers to the 12th day of the 5th month of the 85th year (of some assumed century).

Suppose that you were given the three numerical components of a calendar date, but that you were not told which one identified the month, which one identified the day, and which one identified the year. In that case, there may be multiple ways of ordering the numbers so that they describe a valid calendar date.

For example, if the three numbers were 6, 9, and 25, any of the following would be possible interpretations: `09/25/06`, `06/25/09`, `06/09/25`, and `09/06/25`.

Develop a program that, given three integers in the range 0..99, displays, in chronological order and without duplication, every calendar date in the 21st century (i.e., the years 2000-2099) that those integers could be identifying, in the traditional form exemplified by `September 6, 2025`.

Input: The first line contains a positive integer n indicating how many instances of the problem are described on the succeeding n lines. Each instance of the problem is described on a single line containing three integers i, j, and k, respectively, satisfying $0 \leq i \leq j \leq k \leq 99$.

Output: For each given instance (i, j, k) of the problem, display i, j, and k on one line and then, one per line, display every calendar date (in the 21st century) that can be described using those three numbers, in any order, as month, day, and year. Follow that with a single blank line. The dates should be listed in chronological order (from earliest to latest) with no duplication and should be in the traditional form `<month-name> <day>, <year>`.

Calendar Facts: The months, in order from 1st to 12th, are `January`, `February`, `March`, `April`, `May`, `June`, `July`, `August`, `September`, `October`, `November`, and `December`.

February has 28 days, except during leap years, when it has 29. During the period of concern to us (the years in the range 2000..2099), a leap year is any that is divisible by four. Each of April, June, September, and November has 30 days, and each of the remaining seven months has 31 days.

Sample input and output on next page ...

```
Sample input:
-------------
6
6 9 25
0 2 29
1 2 29
11 31 57
3 4 31
5 5 5



Resultant output:
-----------------
6 9 25
September 25, 2006
June 25, 2009
June 9, 2025
September 6, 2025

0 2 29
February 29, 2000

1 2 29
January 29, 2002
January 2, 2029
February 1, 2029

11 31 57

3 4 31
March 31, 2004
March 4, 2031
April 3, 2031

5 5 5
May 5, 2005
```

Problem 4: Duplicate Characters

Develop a program that, given a string S of characters, produces a string T that contains precisely those characters that appear in S at least twice, in order of their first occurrences in S.

Example: $S =$ `gbaEae#QaegpTaappvXQaap$R` gives rise to $T =$ `gaeQp`

Input: The first line contains a positive integer n indicating how many strings will be provided as input. Each subsequent line contains one such string, which can be assumed to contain only printable ASCII characters (followed by a "newline", of course).

Output: For each string given as input, three lines of output are to be produced. The first line is to display the input string S; the second line is to display the output string T (containing precisely the characters that occur multiple times in S, and in order of their first occurrences in S); the third line is to be blank.

```
Sample input:
-------------
3
gbaEae#QaegpTaappvXQaap$R
abcdefghijkLMNOP
The cat in the hat ate a rat for dinner.


Resultant output:
-----------------
gbaEae#QaegpTaappvXQaap$R
gaeQp

abcdefghijkLMNOP


The cat in the hat ate a rat for dinner.
he atinr
```

Problem 5: Character Counting

Develop a program that, given a string S of characters, produces a collection of strings T_i, $i = 1, 2, \ldots, k$, where k is the maximum number of times that any character appears in S and where each T_i contains precisely those characters that appear in S exactly i times. Moreover, in each T_i the order in which the characters appear must correspond to the order of their first appearance in S.

Input: The first line contains a positive integer n indicating how many strings will be provided as input. Each subsequent line contains one such string, which can be assumed to contain only printable ASCII characters (followed by a "newline", of course).

Output: For each string S given as input, S should be displayed on one line, and on the following lines each of the T_i's that is nonempty should be displayed (with i going from 1 to k), sandwiched between a pair of dollar signs (`$`) and preceded by the value i, a colon, and a space. A blank line should appear at the end.

Sample input and output appear on the next page...

```
Sample input:
-------------
3
The cat and the rat ate some cheese together.
Where, oh where, has my Underdog gone?
Kirk, Spock, and McCoy helped the Mother Horta.


Resultant output:
-----------------
The cat and the rat ate some cheese together.
1: $Tndmg.$
2: $crso$
4: $ha$
6: $t$
8: $ $
9: $e$

Where, oh where, has my Underdog gone?
1: $WwasmyU?$
2: $,ndg$
3: $ro$
4: $h$
6: $e $

Kirk, Spock, and McCoy helped the Mother Horta.
1: $KiSnCylH.$
2: $k,pcadM$
3: $rht$
4: $oe$
7: $ $
```

Problem 6: Triangle Completion

Develop a program that, given as input three positive real numbers a, b, and c (each of which is smaller than the sum of the other two), identifies a triangle whose sides have lengths a, b, and c.

More precisely, the program is to determine the third vertex P_3 of a triangle whose other vertices are $P_1 = (0, 0)$ and $P_2 = (a, 0)$. The y-coordinate of P_3 must be positive, the distance between P_1 and P_3 must be b, and the distance between P_2 and P_3 must be c.

Input: The first line contains a positive integer n indicating how many triangles are to be identified. Each of the next n lines contains three positive real numbers, a, b, and c, separated by spaces. Each number is less than the sum of the other two.

Output: For each triple (a, b, c) provided as input, the program should produce a line of output showing the values of a, b, and c, followed by a colon and a space, followed by (an approximation to) the coordinates of the unique point (x_1, y_1) such that $y_1 > 0$, $d((0, 0), (x_1, y_1)) = b$, and $d((a, 0), (x_1, y_1)) = c$, where

$$d((x_1, y_1), (x_2, y_2)) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

is the distance function. Because floating point arithmetic (as employed by digital computers in manipulating real numbers) generally results in a loss of precision, your results need not match the judges' expectations exactly.

```
Sample input:
-------------
4
2.0 2.0 2.0
3.5 1.0 4.0
5.0 8.0 4.5
10.25 15.4 18.25

Resultant output:
-----------------
2.0 2.0 2.0: (1.0,1.7320508075688772)
3.5 1.0 4.0: (-0.39285714285714285,0.9195995135416952)
5.0 8.0 4.5: (6.875,4.090767042988393)
10.25 15.4 18.25: (0.44682926829268416,15.393516284624415)
```

Problem 7: Decimal to Rational Conversion

A *rational* number is one that can be expressed as a fraction p/q, where p and q are integers and q is nonzero. The rationals also are precisely the numbers that can be expressed in *repeating decimal* form. That is, a number is rational if and only if it can be written in the form

$$p.d_1d_2\cdots d_m\overline{d_{m+1}d_{m+2}\cdots d_{m+r}}$$

where p is an integer, $m \geq 0$, $r > 0$, each d_i is a digit between 0 and 9, and the bar over the last r digits indicates that that sequence of digits repeats forever.

Develop a program that, given as input a repeating decimal representation of a positive rational number in the closed interval $[0, 1]$, produces as output a fraction in simplest form representing that rational number. (A fraction in simplest form is one in which the greatest common divisor of the numerator and denominator is one.)

To illustrate how such a conversion can be done, consider as an example $0.021\overline{35}$:

Let $x = 0.021\overline{35}$. Then $10^3x = 21.\overline{35}$ and $10^5x = 2135.\overline{35}$.

$$\begin{aligned} 10^5x - 10^3x &= 2135.\overline{35} - 21.\overline{35} \\ 99000 \cdot x &= 2114 \\ x &= 2114/99000 \\ x &= 1057/49500 \end{aligned}$$

Input: The first line contains a positive integer n indicating how many instances of the problem are to be solved. Each of the remaining n lines of input contains a repeating decimal number in the form $\mathrm{d_1d_2\ldots d_m\ldots d_{m+r}}$ r, which is to be interpreted to mean

$$0.d_1d_2\cdots d_m\overline{d_{m+1}d_{m+2}\cdots d_{m+r}}$$

That is, we omit the whole number part, which is understood to be zero, and the decimal point; also, we use r to indicate how many digits repeat (because there is no convenient way, using plain text, to place a bar over those digits). For example, $0.083\overline{24}$ would be given by `08324 2` and $0.\overline{9}$ (which is equal to 1.0!) would be given by `9 1`.

Output: For each numeral given as input, it is displayed as it was given, except preceded by `0.`, followed by an equals sign, followed by an equivalent fraction in simplest form. (You may include a space on either side of the slash in the fraction.)

Sample input and output appear on next page...

```
Sample input:
-------------
4
3285714 6
02135 2
0 1
9 1


Resultant output:
-----------------
0.3285714 6 = 23/70
0.02135 2 = 1057/49500
0.0 1 = 0/1
0.9 1 = 1/1
```

Problem 8: Olympic Medal Count

During each Olympic Games, newspapers and TV networks frequently show a "medal count" that lists how many gold, silver, and bronze medals have been won by the athletes of each participating country. Usually, the countries are listed in descending order by how many medals, in total, its athletes have won. Ties are broken first by considering the number of gold medals won; if those numbers are equal, the number of silver medals won is used to break the tie. If two countries match exactly in numbers of gold, silver, and bronze medals won, they are listed in alphabetical order, using their three-letter country code.

Develop a program that, given as input for each Olympic event the countries represented by its gold, silver, and bronze medal winners, displays a medal count, as described above and illustrated below.

Input: The first line contains a positive integer n indicating how many Olympic events have been completed. Each of the next n lines contains the three-letter codes identifying the countries represented by the gold, silver, and bronze medal winners, respectively, of one event.

Output: The output produced should be a table giving the medal counts of each country, as illustrated below. The first line of the table should have column headings `G`, `S`, `B`, and `T` (referring to the three medal colors and to "Total"). Each subsequent line of the table should include a country's three-letter code followed by the numbers of gold, silver, and bronze medals won by athletes from that country, followed by the total number of medals. Countries are to be listed in descending order by total medals won, with ties broken by number of gold medals, then by number of silver medals, and, finally, by country code (alphabetical order). Each number should be right-justified in a field of width three. (Note that it is possible for a medal count to reach two digits.)

Sample input and output appear on next page...

Sample input:

```
------------
15
USA GBR DEU
DEU FRA NLD
NOR NOR USA
RUS USA USA
NLD RUS DEU
NOR POL NOR
CAN RUS USA
NLD RUS POL
SWE CHE RUS
RUS SWE NLD
USA LVA GBR
DNK POL DEU
NOR CAN CAN
NLD ITA USA
JPN AUT AUS
```

Resultant output:

```
-----------------
     G  S  B  T
USA  2  1  4  7
RUS  2  3  1  6
NOR  3  1  1  5
NLD  3  0  2  5
DEU  1  0  3  4
CAN  1  1  1  3
POL  0  2  1  3
SWE  1  1  0  2
GBR  0  1  1  2
DNK  1  0  0  1
JPN  1  0  0  1
AUT  0  1  0  1
CHE  0  1  0  1
FRA  0  1  0  1
ITA  0  1  0  1
LVA  0  1  0  1
AUS  0  0  1  1
```