Consortium for Computing Sciences in Colleges
2024 Southeastern Programming Competition
Saturday, November 2nd, 10 AM – 1 PM EDT

There are nine (9) problems in this packet. 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. Here are some reminders for our competition.

- Each team may comprise 2–4 students, but only up to two students can be at a machine at a time.
- Internet access is only allowed to consult the APIs of a programming language. Cell phone usage is not allowed at any time.
- Any team found to be communicating with someone other than their teammates for assistance will be disqualified.
- Four programming languages are allowed: C#, C++, Java, Python
- An awards ceremony will be held at 1:30 PM EST.
- Have a lot of fun and good luck! 😊

Problem 1

Won't You Be My Neighbor?

In the biography *Won't You Be My Neighbor?* on the life of iconic children's television host Fred Rogers, we learn that the number 143 was his favorite number. Mr. Rogers explained it as a kind of code, "One is 'I,' four ('L,' 'O,' 'V,' 'E'), three ('Y,' 'O,' 'U'): 'I love you.'" We even learn that he tried to keep his weight at 143 pounds for his adult life. The movie explores many of his values and profound life lessons such as always looking out for helpers when a bad or stressful event happens in our lives.

Your job is to write a program that would make Mr. Rogers proud by converting a sentence into a single number based on his code above where the length of each of the words within a sentence is concatenated into a single number. Sum up the codes for each line and print the grand total sum of all the codes for the file.

Input

The input will consist of one or more strings. Each string will consist of 0 to 80 characters on a single line made up solely of spaces and/or letters followed by an end of line marker. You may have up to 40 words of length ≥ 1 on a line with one or more spaces separating each of the words. The end of input is denoted by a line with the string "THE END" in all uppercase at the start.

Output

Keep track of all line numbers and output the code for each line in the format below. This should be followed by a single line with the grand total sum of all line codes for the file as seen below. The sum may be a number of indefinite length.

Sample Input

```
I Love You
Look for the Helpers
        Furman University
Saint Valentine Day
Norway Sweden Finland Denmark Scandanavia
Mary Poppins Returns to Walt Disney World in March
THE END
```

Output Corresponding to Sample Input

```
Line 1 = 143
Line 2 = 4337
Line 3 = 610
Line 4 = 593
Line 5 = 667711
Line 6 = 477246525
Sum of file = 477919919
```

Problem 2

Matrix Order

A row of a matrix is said to be in *ascending order* if all of the elements in that row are arranged from smallest to largest. We want to write a computer program that can calculate the total number of ascending rows in a given 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 representing a different matrix. 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, $2 \leq c \leq 100$ representing the number of columns. This is followed by r rows and c columns of valid integers in that particular matrix. Each integer is separated by a single space. Your input will be in row major order. You may assume there will be no duplicate integers on any row.

Output

For each matrix input, you should print out the number of rows in ascending order for each matrix in the format below. Use a blank line after each line of output.

Sample Input

```
5
2 3
4 6 24
-11 -9 -8
3 4
1 2 3 4
4 5 6 7
7 8 9 10
2 2
5 6
4 1
1 2
8 -9
3 4
1 2 3 4
2 3 4 5
3 4 2 5
```

Output Corresponding to Sample Input

```
Number of Ascending Rows in Matrix #1 = 2

Number of Ascending Rows in Matrix #2 = 3

Number of Ascending Rows in Matrix #3 = 1

Number of Ascending Rows in Matrix #4 = 0

Number of Ascending Rows in Matrix #5 = 2
```

Problem 3

Time Loop Trap

In the mystical land of Chronosia, time flows in mysterious ways. Chronosia is governed by a powerful artifact known as the Chrono Crystal, which controls the flow of time in the land's many towns. Some towns in Chronosia are connected to others by magical pathways, through which time flows from one town to another. Recently, the Chrono Crystal has started malfunctioning, causing time loops that allow citizens to travel in endless cycles, preventing them from moving forward in time. The wise Chronomancer of Chronosia has tasked you, the Guardian of Time, with identifying and breaking these time loops. Your mission is to determine if any of the towns in Chronosia are trapped in a time loop. If there is even a single time loop, you must alert the Chronomancer so that it can be fixed.

You are given a map of Chronosia with n towns and m magical pathways. Each pathway allows time to flow in one direction from one town to another. Your task is to determine whether there is a time loop in the system of towns and pathways. A time loop occurs if there is a sequence of towns $T_1 \rightarrow T_2 \rightarrow \cdots \rightarrow T_k \rightarrow T_1$ such that time flows from T_1 to T_2, from T_2 to T_3, and so on, eventually flowing back to T_1.

Input

The first line contains an integer c, the number of test cases. The first line of the test case contains two integers n and m ($1 \leq n \leq 5000$, $0 \leq m \leq 5000$), representing the number of towns and the number of magical pathways, respectively. The next m lines each contain two integers u and v ($1 \leq u, v \leq n$), representing a magical pathway that allows time to flow from towns u to v.

Output

Output `YES` if there is at least one time loop in Chronosia, otherwise output `NO`.

Sample Input

```
2
4 4
1 2
2 3
3 4
4 2
5 5
1 2
2 3
3 4
4 5
4 5
```

Output Corresponding to Sample Input

```
YES
NO
```

Problem 4

Narcissistic Numbers

Write a program to determine if a given number is narcissistic. A narcissistic number is a number that is the sum of its own digits each raised to the power of the number of digits.

Input

The input will be a single integer c ($0 \leq c \leq 100$), the number of test cases. Then follow c integers c_i , $i = 1, 2, \cdots, c-1, c$, with each c_i , on a line by itself and $0 \leq c_i \leq 10^{15}$.

Output

For each value c_i , write on a single line the case number, and YES if c_i is a narcissistic number, NO otherwise. See sample below.

Sample Input

```
4
1
371
22
28116440335967
```

Output Corresponding to Sample Input

```
Case #1: YES
Case #2: YES
Case #3: NO
Case #4: YES
```

Problem 5
Leap Day

2024 is a leap year, and we had an extra day in February of this year. As you may know, this extra day keeps our Gregorian calendar year from drifting through the seasons over time since a year is technically 365.25 days. The Gregorian reform of 1582 modified the old Julian calendar's scheme of leap years 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 years 1700, 1800, and 1900 are not leap years, but the years 1600 and 2000 are.

Your job is to write a computer program which will calculate the total leap years in an inclusive range of years.

Input
The input will start with a single line containing a positive integer *n* representing the specified number of year ranges to be input. Each of the *n* line(s) consist of two integers starting with the first year, a single space, and followed by the second year. You may assume all years fall in the range [`1600`, `2100`], and that the second year is always larger than the first year.

Output
Keep track of all line numbers and output the code for each line in the exact format below. If there is only one leap year, use the verb `is` rather than `are` in the output along with the singular word `year` rather than `years`. Make sure you end the output of each line with a period.

Sample Input
```
5
2015 2024
2015 2016
2014 2015
1900 2000
1900 2100
```

Output Corresponding to Sample Input
```
There are 3 leap years from 2015 to 2024.
There is 1 leap year from 2015 to 2016.
There are 0 leap years from 2014 to 2015.
There are 25 leap years from 1900 to 2000.
There are 49 leap years from 1900 to 2100.
```

Problem 6

Swimming at the 2024 Paris Olympics

The Summer Olympics 2024 were held in Paris, where swimmers from various countries competed in different swimming events. Each event comprised multiple heats, and each heat had a list of swimmers with their recorded times.

Your task is to determine each event's gold, silver, and bronze winners based on the best times across all heats. Each event can have multiple heats, and each heat can have a variable number of swimmers.

Output the swimmers who win each event's gold, silver, and bronze medals. If two or more swimmers have the same time, the swimmer who appears first in the input should be considered to have a better rank. If there are not enough swimmers to award all three medals, only output the medals that can be awarded.

Input

The input starts with a single integer E ($1 \leq E \leq 100$), representing the number of events.
For each event:

- The first line contains a single integer H ($1 \leq H \leq 10$), representing the number of heats in that event.

For each heat:

- The first line contains a single integer N ($1 \leq N \leq 100$), representing the number of swimmers in that heat.
- The next N lines each contain a string S ($1 \leq |\ length\ of\ S\ | \leq 50$) representing the swimmer's last name with no spaces, a string C representing the nationality of the swimmer using three letters), and a floating-point number T ($0.0 \leq T < 1000.0$), representing their recorded time in seconds.

Output

For each event, output three lines in the following format followed by a blank line:

Event X: Gold - S (C)
Event X: Silver - S (C)
Event X: Bronze - S (C)

Where X is the event number (starting from 1), S is the swimmer's name, and C is their nationality. Only output the medals that can be awarded based on the number of swimmers.

Sample Input

```
1
2
2
Steve USA 59.23
John CAN 58.95
1
Frank GBR 57.60
```

Output Corresponding to Sample Input

```
Event 1: Gold - Frank (GBR)
Event 1: Silver - John (CAN)
Event 1: Bronze - Steve (USA)
```

Problem 7

Monkeying Around

The infinite monkey theorem states that a monkey hitting keys at random on a typewriter keyboard for an infinite amount of time will almost surely type any given text, including the complete works of William Shakespeare. Now, let us apply this to one of my favorite activities, Christmas-treeing multiple choice tests. Christmas-treeing a test refers to a random answer selection strategy for Multiple Choice Scantron style tests.

Suppose we have a multiple choice test, where each question has *o*, a positive integer less than six options. The input argument will give the number of options for each question (say, if the exam had four options, *a*, *b*, *c*, *d* then *o* = 4), before giving the true (the teacher's) answer key for the exam as one long string. For instance, a sample input could look like: `4,abbabcabbacad`

Let *x* be the minimum number of times you would need to Christmas-tree an exam before there was over a 50% chance one of my attempts matched the teacher's answer key. Return the remainder of *x* divided by 1,000,000. Note that whenever you generate an answer key and get it wrong, you ignore that and randomly generate another answer key as a guess. You could perhaps generate the same answer key twice. This is more akin to monkeys "typing Shakespeare."

Input

The first line of input contains a single integer *n*, ($1 \leq n \leq 100$), the number of test cases that follow. For each test case, the sample input gives a positive integer *o*, ($1 \leq o \leq 5$), followed by a comma, and then a string *S* of letters representing the answer key, ($1 \leq |\, length\ of\ S\, | \leq 20$).

Output

For each line of input, output the minimum number of answers *x* such that the probability of having at least one incorrect answer is less than 50%. If the answer key contains more unique characters than the number of options *o*, output `Impossible Answer Key!`

Sample Input

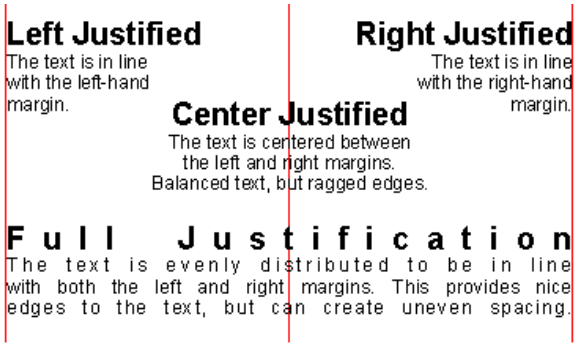
```
8
1,ab
1,aa
2,aba
2,aaa
3,aba
3,abc
4,abc
4,abbabcabbacad
```

Output Corresponding to Sample Input

```
Impossible Answer Key!
1
6
6
19
19
45
516320
```

Problem 8

Word Gaps



Sometimes you want your text to look *flush*. That is, to have full justification so that it is aligned along both the left and margins. To make this happen, extra spaces have to be added between words as necessary. We want to write a program to calculate the number of words on a line, the total characters in those words, and the basic gap width between those words that would produce flush text. Any leftover spaces are inserted one at a time into the gaps from left to right until there are no more leftover spaces. Assume each formatted string will be of length 80 after all those gaps are filled in with spaces. The gap width is equal to :

(80 – Total Characters in all Words on Line) / (Total Words on Line – 1)

Input

The first line of input contains a single integer *n*, ($1 \leq n \leq 1000$), the number of test cases that follow. Each test case consists of one line of input containing a string of length *n*, ($1 \leq n \leq 80$). This string will consist of two or more words as input. A word is defined as one or more contiguous characters followed by white space (space, tab, or newline).

Output

For each test case, output it exactly as formatted below showing the line number, followed by a sing colon, single space, total words on the line, total characters in all words on the line, and the gap width. Use integer division for all gap width calculations, and note the denominator will never be 0 since each line will always have at least two words.

Sample Input

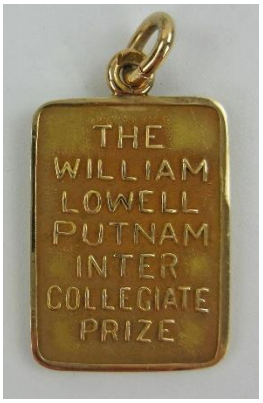
```
5
When April with his
sweet showers has
pierced the drought of March
to the root.
The End
```

Sample Output

```
Line 1: words=4 chars=16 gap width=21
Line 2: words=3 chars=15 gap width=32
Line 3: words=5 chars=24 gap width=14
Line 4: words=3 chars=10 gap width=35
Line 5: words=2 chars=6 gap width=74
```

Problem 9

Completing the Binomial



The 79th William Lowell Putnam Mathematical Competition held in December 2018 posed the following question: Find all ordered pairs (a,b) of positive integers for which $1/a + 1/b = 3/2018$

Your task is similar. For a given ratio p/q (where p is prime), find all solutions (a,b), $a \leq b$, to

$$1/a + 1/b = p/q.$$

Since this is not a math competition, you can use the equivalent equation:

$$(pa - q)(pb - q) = q^2.$$

However, since your solution will be run on a computer, your solution needs to handle larger values of q, $q \leq 99999999$.

Input

The first line will contain n, $n \leq 20$, the number of cases. The next n lines will contain a prime p followed by a positive integer q where $p \leq q$, p and q have no factor in common, and $q \leq 99999999$.

Output

For each case, print one line with the case number and text as shown in the example.
Next, for each solution (a,b), $a \leq b$, print a and b separated by a space. The solutions should be sorted by the a's. Also, separate each case by a blank line.

Sample Input

```
2
3 2018
17 20241102
```

Sample Output

```
Case 1: 1/a + 1/b = 3/2018
673 1358114
674 340033
1009 2018

Case 2: 1/a + 1/b = 17/20241102
1190657 359704623642
1190658 286907500299
1190679 54650011538
1190724 19984714708
1208424 80964408
1212834 65103843
1307103 13364594
1445793 6747034
```