This problem set contains 10 questions (A-J)

22nd September 2013

Jointly organized by
Department of Computer Science and Information Sciences
University Teknologi Petronas

&

Kulliyyah of Information and Communication Technology
International Islamic University Malaysia
as ACM ICPC Malaysia Office

Judges:

1. Mdm. Marini Abu Bakar
2. Dr. Rohiza Bt. Ahmad
3. Dr. Rizal Mohd. Nor
4. Dr. M. Nordin B. Zakaria
5. AP. Dr. Syed Ahmad Aljunid
6. AP. Dr. Normaziah Abdul Aziz
7. AP. Dr. Muthukkaruppan Annamalai
8. Prof. Dr. Teddy Mantoro (Chief Judges)
9. Prof. Dr. Zarina Shukur

Problem Set Contributors

Problem		Contributor	University
A	Trailing Sifar	Mdm. Marini	UKM
B	Let's Go Green	Prof. Dr. Zarina	UKM
C	Filter & Perform	Dr. Rohiza	UTP
D	Show me the Step	Dr. Rohiza	UTP
E	The Sultan's Chapati	Dr. Rizal	IIUM
F	Sahur & Imsa	AP. Dr. Normaziah	IIUM
G	Rice Sack	AP. Dr. Normaziah	IIUM
H	Rain Forest Canopy	AP. Dr. Muthu	UiTM
I	Positive Closure Sum	Prof. Dr. Teddy	USBI, Jakarta
J	LIS	Mr. Risan	NTU, Singapore

Problem set level

Problem		Level
A	Trailing Sifar	Medium
B	Let's Go Green	Medium
C	Filter & Perform	Easy
D	Show me the Step	Easy
E	The Sultan's Chapati	Medium
F	Sahur & Imsa	Easy
G	Rice Sack	Easy
H	Rain Forest Canopy	Medium
I	Positive Closure Sum	Difficult
J	LIS	Difficult

A	TRAILING *SIFAR*	
	Input	Standard Input
	Output	Standard Output
	Time Limit	10 seconds

Problem Description

Sifar is a malay word for zero or 0. In mathematics, trailing *sifar* is a sequence of 0s in the decimal representation of a number after which no other digits follow.

The number of trailing *sifar* in the decimal representation of ***N***! (5 <= ***N*** <= 1,000,000) is simply the multiplicity of the prime factor 5 in ***N***!. Given a decimal integer ***N***, you are to find the number of trailing *sifar* for ***N***! For example, 10! = 3,628,800. Thus, the number of trailing *sifar* for 10! is 2.

Input

Each line of input contains an integer ***N*** where 5 <= ***N*** <= 1,000,000. The input is terminated with a line containing 0.

Output

For each test case, the output contains a line in the format Case #x: ***M***, where x is the case number (starting from 1) and ***M*** is the number of trailing *sifar* for ***N***!

Example of sample I/O

Sample Input	Sample Output
5 10 118	Case #1: 1 Case #2: 2 Case #3: 27

211 0	Case #4: 51

B IT'S NOT WHAT YOU KNOW BUT WHO YOU KNOW

Input	Standard Input
Output	Standard Output
Time Limit	10 seconds

Problem Description

Khairy is a young, smart and highly talented politician. However, he still believes that for success, and especially to obtain something, one's network of personal contacts is essential. Next month will be the Supreme Council election of his political party. He wants to contest and really hope to win in the election. Therefore, support from top influential politician is needed.

Time is running out, therefore he plans a strategy. If he cannot meet the top person directly he will use other politicians to introduce him to this top leader. In order to do this, he models the strength of relationship among the politicians.

The strength of relationship between two people is represented as follows: Faithful (1), Close (2), Buddies (3) and Acquaintance (4). Between two politicians, either one of them will normally stimulate the communication. No representation for enemy. It will be much better if a faithful friend introduces Khairy to someone rather than an acquaintance. So Khairy wants to minimize the sum of strength of the relations he is using to meet the top influential politician.

From a list of N politicians, given their relationship strengths, find a list of politicians so that Khairy meets the top influential politician through them while the sum of strength of all the relationships he used is minimized.

Task

Your task is to come out with a sequence of politicians that Khairy needs to meet in order to get introduced to the top influential politician.

Input

The first line of input is **T** which is the number of cases, 1 <**T**< 1000. This is followed by the test cases. Each case consists of several lines. The first line contains 2 numbers, first is **N** which represents the number of relationships, **N** ≤ 200, and **M** which represents the number of politicians, 5 ≤**M**≤ 20. Each of the next **N** lines contains three integers that represent politician **x**, his/her friend **y** (0 ≤**x**,**y**< **M**) and the strength of relationship **z** (1 ≤**z**≤ 4**)**. Politician x=0 represents Khairy and politician x=M-1 represents the top politician.

Output

For each test case, the output contains a line in the format **Case #**x: where x is the case number (starting from 1) followed by a colon, followed by a sequence of politicians to meet. If there is multiple valid sequence of politicians print any of them. The list must start with 0 (Khairy's id) and end with M-1, the top politician. If there is no way Khairy can meet the to politician, print -1.

Sample Input	Sample Output
2 7 5 0 1 2 1 3 4 0 2 1 0 4 3 3 2 3 3 4 4 2 4 1 3 5 0 1 2 1 3 4 4 2 1	Case #1: 0 2 4 Case #2: -1

Explanation for the first case:
Khairy can ask politician 1 (strength 2, close) to introduce him to politician 3 (strength 4, acquaintance), who can finally introduce Khairy to politician 4 (strength 4, acquaintance). In that way total strength used in this situation is 2+4+4 = 10.

Khairy can straightly talk to politician 4 (strength 3, buddy). But instead if Khairy asks politician 2 (strength 1, faithful) to introduce him to politician 4 (strength 1, again faithful) – the total strength used

in this situation is 2, which is better than the other two situations and Khairy is more likely to make a good impression.

C	FILTER AND PERFORM	
	Input	Standard Input
	Output	Standard Output
	Time Limit	10 seconds

Problem Description

A data set in an input file contains several lines of data. The first line provides two information, the first is an integer between 1 to 5 (inclusive), which represents the number of subsequent lines in the data set and the second is any of the following symbols {+, -, ^}, which represents operation to be performed. The second line onwards has one data element each which can be of positive integer, positive real number or string (with maximum length of 10). A program is to be written so that if the symbol in the first line is a '+', then only the lines containing integers in the data set will produce output. The output will be the sum of each digit in the integer data. On the other hand, if the symbol is a '-', then only the lines containing strings will produce output. The output will be the string with all character 'a' (if any) being removed. If the symbol is a '^', then only the lines containing real numbers will produce output. The output will be the number in its exponential format. For example 2.53 will be represented as 2.53e0, 25.533 will be 2.5533e1 and 0.253 will be 2.53e-1.

Input

The first line of the input data contains an integer which represents the number of test cases. The line is then followed by the data for each of the test cases. Each test case begins with a line containing two data, i.e., an integer n denoting the number of data lines ($1 \le n \le 5$) and a special symbol '+', '-', or '^' denoting operation to be performed. Each data line in the data set contains either a positive integer, a positive real number or a string with maximum size of 10.

Output

The output comprises of one line for each test case. The line begins with prefix "Case #x:" where x represents the case number, followed by the output of the test case. Depending on the symbol provided in the first line of the test case, only data of applicable type will produce output with a space in between.

Sample Input	Sample Output
3 3 + 567 2.53 10 4 - 2.53abc DAf aag 123 3 ^ 253.3 Def 0.00067	Case #1: 18 1 Case #2: 2.53bc DAf g Case #3: 2.533e2 6.7e-4

D SHOW ME THE OPERATORS

Input	Standard Input
Output	Standard Output
Time Limit	10 seconds

Problem Description

A programming instructor needs your help to teach his students on precedence level of binary arithmetic operators (+, -, *, /). Write a program which can show the sequence of operators executed when evaluating an arithmetic expression according to the precedence rules below.

Precedence rules:

(1) Multiplication and division operators have higher precedence than plus and minus operators
(2) Multiplication and division operators have the same level of precedence. However, in an expression, the operator which appears first from the left of the expression has higher precedence.
(3) Plus and minus operators have the same level of precedence. However, in an expression, the operator which appears first from the left of the expression has higher precedence.

Evaluation of expression starts from left to right.

Input

The first line of the input data contains an integer which represents the number of test cases. The line is then followed by the data for each of the test cases. Each test case contains an arithmetic expression containing mixture of positive integer operands and binary arithmetic operators (with a space in between). For simplicity, let's assume the maximum number of operators in an expression is limited to 10.

Output

The output comprises of one line for each test case (arithmetic expression). The line begins with prefix "Case #x:" where x represents the case number, followed by the output of the test case. For each test case, a list of arithmetic operators will be produced according to the precedence they were executed in the expression. The list is terminated with the final result of the execution.

Sample Input	Sample Output
2 4 + 16 / 4 * 3 + 6 * 2 6 * 2 + 3 * 2 / 2	Case #1: / * * + + 28 Case #2: * * / + 15

E THE SULTAN'S CHAPATI

Input	Standard Input
Output	Standard Output
Time Limit	2 seconds

Problem Description

The Sultan of Isketambola likes his Chapati served to him in a stack of uniquely sized Chapatis and heated up in a special way. The cook for the Sultan would have to heat up a stack of uniquely sized Chapati where each Chapati on the bottom of the stack is larger in diameter to the one above. You are to write a program that indicates how the stack can be sorted so that the largest Chapati is on the bottom and the smallest Chapati is on the top. The size of a Chapati is given by the Chapati's diameter. All Chapatis in a stack have different diameters.

Sorting a stack is done by a sequence of Chapati "flips". A flip consists of inserting a spatula between two Chapatis in a stack and flipping (reversing) the Chapatis on the spatula (reversing the sub-stack). A flip is specified by giving the position of the Chapati on the bottom of the sub-stack to be flipped (relative to the whole stack). The Chapati on the bottom of the whole stack has position 1 and the Chapati on the top of a stack of *n* Chapatis has position *n*.
A stack is specified by giving the diameter of each Chapati in the stack in the order in which the Chapatis appear.

For example, consider the three stacks of Chapatis below (in which Chapati 8 is the top-most Chapati of the left stack):

8	2	4
6	4	2
1	1	1
4	6	6
2	8	8

The stack on the left can be transformed to the stack in the middle via *flip(1)*. The middle stack can be transformed into the right stack via the command *flip(4)*. A final flip of flip(3) will result in a sorted stack.

Input

First line of the input contains T the number of test cases (1<=T<=1000). Each test case consists two lines. First line of the test case contains N (1<=N<=30), the number of Chapatis in the stack. The next line contains N integers separated by a single space. Each integer specifies the diameter of the Chapati between 1 and 100 from the top most position to the bottom.

Output

For each test case, the output contains a line in the format Case #x: followed by a sequence of integers, where x is the case number (starting from 1). For each stack the sequence of flips should be terminated by a 0 (indicating no more flips are necessary). Once a stack is sorted, no more flips should be made.

Example of sample I/O

Sample Input	Sample output
3 5 1 2 3 4 5 5 5 4 3 2 1 5 8 6 1 4 2	Case #1: 0 Case #2: 1 0 Case #3: 1 4 3 0

F	SAHUR & IMSA'	
	Input	Standard Input
	Output	Standard Output
	Time Limit	1 second

Problem Description

Midhat is a Network Security Engineer, based in Sarajevo. He is assigned to do some important consultation projects around the globe in July and August 2013. It happened that Ramadhan (the fasting month for Muslim) falls during these months for the year 2013. Midhat has to travel to several cities – Istanbul, Kuala Lumpur, Tokyo, Melbourne, Sao Paolo and Chicago. Even though it is permissible for Muslims not to fast when travelling, he prefers to continue fasting in Ramadhan. Midhat has no problem on this matter except he needs to make himself awake for *sahur* (early breakfast before dawn) on his own, which he is used to be waked by his mother at home.

Midhat wants to have his *sahur* 45 minutes before *imsa'* (end time for taking *sahur*). He decided to set his travelling alarm clock 45 minutes before imsa', so that he can take his *sahur* in time, make his *Fajr* prayer and ready to work early. Since he's travelling to different parts of the world, the *imsa'* time differs from one city to another. Midhat wants to set his travelling clock to wake him up on time for all the cities he visits. Help Midhat by writing a program that will take one time stamp, in 24-hour notation, and print out a new time stamp, 45 minutes earlier, also in 24-hour notation.

Note: In 24-hour time notation, it starts with 0:00 (midnight) and ends with 23:59 (one minute before midnight). In the input and output we'll ignore the leading zeros and colon for simplicity. So 0:00 will be written as 0 0.

Input

The first line will contain number of test cases, **T**. After that **T** lines will follow, where each line will contain exactly two integers **H** and **M** ($0 \le \mathbf{H} \le 23$, $0 \le \mathbf{M} \le 59$) separated by a single space, the input time in 24-hour notation. **H** denotes hours and **M** minutes.

Output

For each test case, the output contains a line in the format Case #x: followed by a sequence of integers, where x is the case number (starting from 1) and output one line with exactly two integers, the time 45 minutes before input time.

Sample Input	Sample Output
4 5 0 10 10 0 30 23 47	Case #1: 4 15 Case #2: 9 25 Case #3: 23 45 Case #4: 23 2

G	RICE SACK	
	Input	Standard Input
	Output	Standard Output
	Time Limit	1 second

Problem Description

Several sacks of rice need to be transported to five Orphanage Houses. The heaviest sack will go to Orphanage House Al-Ameen because it has the most number of orphanges. The lightest will be sent to Orphanage House Mutiara due to the small number of children staying there.
Given a row of rice sacks, decide which sack goes to Al-Ameen?

Input

The first line is an integer that represent the number of case. The following lines have 5 integers indicating the weights of 5 rice sacks, each separated by a blank. No sack will have a weight of more than 100 unit.

Output

For each test case, the output contains a line in the format Case #x: followed by a sequence of integers, where x is the case number (starting from 1) and an integer that indicates the weight of a rice sack that will go to Al-Ameen.

Sample Input Output

Sample Input	Sample Output
4 1 6 10 5 20 5 10 25 3 1 30 15 5 1 8 7 4 20 50 5	Case #1: 20 Case #2: 25 Case #3: 30 Case #4: 50

H	RAINFOREST CANOPY	
	Input	Standard Input
	Output	Standard Output
	Time Limit	10 seconds

Problem Description

A tropical rainforest is typically divided into four main layers: the emergent, canopy, understory, and forest floor layers. Of the four, it is quite tricky to distinguish the emergent layer from the canopy layer. The emergent layer contains a small number of very large trees, which grow above the general canopy, reaching heights of 45 to 55 metres. The canopy layer is the richest layer of the diverse rainforest, and ranges in heights of 30 to 45 metres tall. Because of bio-diversity crisis, monitoring rainforest canopy is the mission of Whole Forest Observatory, which is trying to identify suitable canopy research sites.

A group of engineers have found a simple satellite imagery technique that can mark canopy layers on a spot image of a wide rainforest area. The spot image is a square and is stored as pixels, i.e., small cells containing either a `1` or a `0`. Each pixel carries some information about a particular 1 km^2 region. A pixel location contains a `1` if part or its entire represented region is a canopy layer and a `0` if otherwise.

The following assumptions hold:

- A canopy layer is represented by at least a single `1`.
- Cells with adjacent sides on a common pixel that contains a `1`, comprise the canopy layer. A single large canopy layer image will contain all `1`'s.
- Distinct canopy layers do not touch one another.
- There is no wrap-around, i.e., the pixels on the bottom are not adjacent to the top, and the left is not adjacent to the right.

Write a program that reads spot images and correctly counts the number of canopy layers in these images.

Input

The input consists of up to `1000` test cases. Each test case describes a spot image of a rainforest area, which starts on a new line with a positive integer **`N`**, `(1 <= N <= 40)` indicating the dimension of the image. The next **`N`** lines of each test case delineate the pixelated representation of the image.

Output

For each test case, produce a single line of output that starts with the prefix "`Case #x:`" where `x` represents the case number (starting from one and incrementing at each new test case), followed by a single space, and then the result, i.e., the number of canopy layers in the spot image.

Example of sample I/O

Sample Input	Sample Output
6 100100 001010 000000 110000 111000 010100 8 01100101 01000001 00011000 00000010 11000011 10100010 10000001 01100000	Case #1: 3 Case #2: 6

I EVEN POSITIVE CLOSURE SUM

Input	Standard Input
Output	Standard Output
Time Limit	4 seconds

Problem Description

Positive Closure or Kleene Closure can be described as the set of finite-length strings that can be generated by concatenating arbitrary elements of set of strings allowing the use of the same element multiple times. In case of numbers, in short, it is a possible numbers generated. Given the number of available even digit of 1 to 10, sum all positive closures from those digits.
For example,

Even Digit	2	4	6	8
Frequency	2	1	0	0

It means that we can use up to two digits of 2 and one digit of 4. There are exactly 8 distinct numbers that can be constructed using the above digits: 2, 4, 22, 24, 42, 224, 242, 422. The sum of all those numbers is 982.

Input

The first line of input contains an integer T ($T \leq 500$) denoting the number of testcases. Each testcase contains nine (not four) integers P_i ($0 \leq P_i \leq 9$) denoting the number of i-th digit for i = 1..9.

Output

For each test case, the output contains a line in the format Case #x: M, where x is the case number (starting from 1) and M is the output in a single line the sum of all possible numbers generated from the available digits. Modulo the output with 1,000,000,007.

Sample Input	Output for Sample Input
3 0 2 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 1 0 1 0 1 0	Case #1: 982 Case #2: 2 Case #3: 147320

J	LIS	
	Input	Standard Input
	Output	Standard Output
	Time Limit	2 seconds

Problem Description

Mr. C is interested with Longest Increasing Subsequence problem. Given a sequence $S = s_1, s_2, \ldots, s_N$. The Longest Increasing Subsequence is the subsequence $L = l_1, l_2, \ldots, l_k$ of S such that $l_1 < l_2 < \ldots < l_k$.

Given a sequence S, find the total length of LIS of every consecutive subsequence (subsequence which elements are consecutive in the original sequence) of S with non zero length!

Input

The first line of input consists of an integer T denotes the number of cases. It is followed by T blocks, each representing a case.

The first line of each case contains an integers: N ($1 \le N \le 500$), the length of S.

The next N lines each consists of an integer s_i ($1 \le s_i \le N$) denoting the i-th element of S. Each element of S is unique.

Output

Output consists of T lines, each describes the solution for each case with the same order as in input.

Each case consists of a single line with the format "Case #i: S", where i represents the case number and S represents the total length of LIS of every consecutive subsequence of S.

Example of sample I/O

Sample Input	Sample Output
1 3 3 1 2	Case #1: 8