Hosted by University of Computer Studies, Yangon, Myanmar

acm International Collegiate Programming Contest

IBM event sponsor

2016 ACM-ICPC Asia-Yangon Regional Programming Contest

Problem Set

Please check that you have 12 problems and 19 sheets.
(excluding additional materials)

Problem A	**Evaluating Fully Parenthesized Expression**	1 page
Problem B	**The Election**	2 pages
Problem C	**Finding the Determinant**	1 page
Problem D	**Bakery Delivery Scheduler**	1 page
Problem E	**Tree Pendant**	2 pages
Problem F	**Chocolates**	1 page
Problem G	**Vampire Numbers**	1 page
Problem H	**Like Father Like Son**	2 pages
Problem I	**Clustering**	1 page
Problem J	**Green Frog and Ordering**	1 page
Problem K	**One Time Pad**	1 page
Problem L	**Passwords for Sweethearts**	1 page

Note: The input and output for all the problems are standard input and output.

December 9, 2016

Welcome to the 2016 ACM-ICPC Asia-Yangon Regional Programming Contest. Before you start the contest, please be aware of the following notes:

The Contest

1. There are twelve (12) problems in the packet, using letters A-L. These problems are NOT necessarily sorted by difficulty. As a team's solution is judged correct, the team will be awarded a balloon. The balloon colors are as follows:

Problem	Problem Name	Balloon Color
A	Evaluating Fully Parenthesized Expression	Lime
B	The Election	Gray
C	Finding the Determinant	Blue
D	Bakery Delivery Scheduler	Green
E	Tree Pendant	Gold
F	Chocolates	White
G	Vampire Numbers	Red
H	Like Father Like Son	Orange
I	Clustering	Indigo
J	Green Frog and Ordering	Yellow
K	One Time Pad	Purple
L	Passwords for Sweethearts	Pink

2. Solutions for problems submitted for judging are called runs. Each run will be judged. The judges will respond to your submission with one of the following responses. In the event that more than one response is applicable, the judges may respond with any of the applicable responses.

Response	Explanation
Yes	Your submission has been judged correct.
Wrong	Your submission generated output that is not correct.
Output Format Error	Your submission's output is not in the correct format or is misspelled.
Incomplete Output	Your submission did not produce all of the required output.
Excessive Output	Your submission generated output in addition to or instead of what is required.
Compilation Error	Your submission failed to compile.
Run-Time Error	Your submission experienced a run-time error.
Time-Limit Exceeded	Your submission did not solve the judges' test data within 30 seconds.
Other-Contact Staff	Contact your local site judge for clarification.

3. A team's score is based on the number of problems they solve and penalty points, which reflect the amount of time and number of incorrect submissions made before the problem is solved. For each problem solved correctly, penalty points are charged equal to the time at which the problem was solved plus 20 minutes for each incorrect submission. No penalty points are added for problems that are never solved. **Teams are ranked first by the number of problems solved and then by the fewest penalty points.**

4. This problem set contains sample input and output for each problem. However, the judges will test your submission against longer and more complex datasets, which will not be revealed until after the contest. **Your major challenge is designing other input sets for yourself so that you may fully test your program before submitting your run.** You should receive a judgment stating that your submission was incorrect, you should consider what other datasets you could design to further evaluate your program.

5. In the event that you feel a problem statement is ambiguous or incorrect, you may request a clarification. Read the problem carefully before requesting a clarification.
   If a clarification is issued during the contest, it will be broadcast to all teams.
   If the judges believe that the problem statement is sufficiently clear, you will receive the response, "No response, read problem statement."
   If you receive this response, you should read the problem description more carefully. If you still feel there is an ambiguity, you will have to be more specific or descriptive of the ambiguity you have found.
   You may not submit clarification requests asking for the correct output for inputs that you provide, e.g., "What would the correct output be for the input ...?" Determining that is your job unless the problem description is truly ambiguous. Sample inputs may be useful in explaining the nature of a perceived ambiguity, e.g., "There is no statement about the desired order of outputs. Given the input: . . . , would both this: . . . and this: . . . be valid outputs?".

6. Runs for each particular problem will be judged in the order they are received. However, it is possible that runs for different problems may be judged out of order. For example, you may submit a run for problem B followed by a run for problem C, but receive the response for C first.
   Do not request clarifications on when a response will be returned. If you have not received a response for a run within 30 minutes of submitting it, **you may have a runner ask the local site judge to determine the cause of the delay**. Under no circumstances should you ever submit a clarification request about a submission for which you have not received a judgment.
   If, due to unforeseen circumstances, judging for one or more problems begins to lag more than 30 minutes behind submissions, a clarification announcement will be issued to all teams.
   This announcement will include a change to the 30 minute time period that teams are expected to wait before consulting the site judge.

7. The submission of abusive programs or clarification requests to the judges will be considered grounds for immediate disqualification.

8. The submission of code deliberately designed to delay, crash, or otherwise negatively affect the contest itself will be considered grounds for immediate disqualification.

Your Programs

9. All solutions must read from standard input and write to standard output. In C this is ***scanf/printf***, in C++ this is ***cin/cout***, and in Java this is ***System.in/System.out***. The judges will ignore all output sent to standard error (***cerr*** in C++ or ***System.err*** in Java). You may wish to use standard error to output debugging information. From your workstation you may test your program with an input file by redirecting input from a file:

   program < file.in

10. Unless otherwise specified, all lines of program output
    - must be left justified, with no leading blank spaces prior to the first non-blank character on that line,
    - must end with the appropriate line terminator (\n, endl, or println()), and
    - must not contain any blank characters at the end of the line, between the final specified output and the line terminator.

    You must not print extra lines of output, even if empty, that are not specifically required by the problem statement.

11. Unless otherwise specified, all numbers in your output should begin with the minus sign (-) if negative, followed immediately by 1 or more decimal digits. If the number being printed is a floating point number, then the decimal point should appear, followed by the appropriate number of decimal digits. For output of real numbers, the number of digits after the decimal point will be specified in the problem description (as the "precision").
    All floating point numbers printed to a given precision should be rounded to the nearest value. For example, if 2 decimal digits of precision is requested, then 0.0152 would be printed as "0.02" but 0.0149 would be printed as "0.01".
    In simpler terms, neither scientific notation nor commas will be used for numbers, and you should ensure that you use a printing technique that rounds to the appropriate precision.

12. All input sets used by the judges will follow the input format specification found in the problem description. You do not need to test for input that violates the input format specified in the problem.

13. All lines of program input will end with the appropriate line terminator (e.g., a linefeed on Unix/Linux systems, a carriage return-linefeed pair on Windows systems).

14. If a problem specifies that an input is a floating point number, the input will be presented according to the rules stipulated above for output of real numbers, except that decimal points and the following digits may be omitted for numbers with no non-zero decimal portion. Scientific notation will not be used in input sets unless a problem explicitly allows it.

15. Every effort has been made to ensure that the compilers and run-time environments used by the judges are as similar as possible to those that you will use in developing your code. With that said, some differences may exist. It is, in general, your responsibility to write your code in

a portable manner compliant with the rules and standards of the programming language. You should not rely upon undocumented and non-standard behaviors.

(a) One place where differences are likely to arise is in the size of the various numeric types. Many problems will specify minimum and maximum values for numeric inputs and outputs. You should write your code with the understanding that, on the judges' machines:

- A C++ int, a C++ long, and a Java int are all 32-bits wide.
- A C++ long long and a Java long are 64-bits wide.
- A float in both languages is a 32-bit value capable of holding 6-7 decimal digits, though many library functions will be less accurate.
- A double in both languages is a 64-bit value capable of holding 15-16 decimal digits, though many library functions will be less accurate.

The data types on your own machines may differ in size from these, but if you follow the guidelines above in choosing the types to hold your numbers, you can be assured that they will suffice to hold those values on the judges' machines.

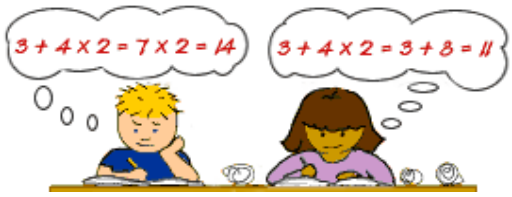
(b) Another common source of non-portability is in C/C++ library structures. Although the C & C++ standards are very explicit about which header file must declare certain std symbols, the standards do not prohibit other headers from duplicating or loading extra symbols.

For example, if your program uses both ***cout*** and ***ifstream***, you might find that your code compiles if you only ***#include <fstream>***, because, as it happens, on your machine the ***fstream*** header ***#includes*** the ***iostream*** header where ***cout*** is properly declared. However, you cannot rely upon the judges' machines having libraries with the same structure. So it is your responsibility to ***#include*** the appropriate headers for whatever std library features you use.

Good luck, and HAVE FUN!!!

Problem A: Evaluating Fully Parenthesized Expression

Little Mary hates Mathematics. Today, she has to do math homework to evaluate the arithmetic expressions. Since she is very poor at mathematics, she doesn't know which operator should be evaluated first in the expression. Her mother helps her by rewriting the expression into fully parenthesized arithmetic expression to show the order of operation as shown in following:



the expression "3+4" is rewritten to (3+4)
the expression "(3+4)*(5-7)" is rewritten to ((3+4)*(5-7))
the expression "9+6*(8-5)" is rewritten to (9+(6*(8-5)))
the expression "9-(-5)/2" is rewritten to (9-((-5)/2))
the expression "9-5/(8-3)*2+6" is rewritten to ((9-((5/(8-3))*2))+6)

But Mary is confused by the fully parenthesized expressions and ask you to write a program for her to evaluate those fully parenthesized expressions.
Your task is to output the evaluated result of those arithmetic expressions.

Input:
The first line contains an integer T(1<=T<=10) which is the number of test cases. The input for each test case is given a single string S representing the expression. S will always contain a valid expression and will be strictly less than 200 characters in length. The operators used in S are '+' symbol for addition, '-' symbol for subtraction and unary minus, '*' symbol for multiplication and '/' symbol for division. All operators (+, - , *, /) are binary and '-' can also be used as an unary operator. All input strings will consist of only characters in the set "0123456789+-*/( )" and there is no white space between characters. The expression should be in correctly fully parenthesized.

Output:
For each test set, a real number or 'Infinity' is printed as shown in sample output. The real number represents the result of evaluation of corresponding expression and 'Infinity' is printed if the expression includes 'Division by 0'.

Sample Input	Sample Output
8	324.0
(310+14)	11.5
(9-((-5)/2))	136.0
((123+(4*5))-7)	13.0
((9-((5/(8-3))*2))+6)	Infinity
(5+(4/(2+(-2))))	9.5
((3+((3*(5+4))/2))-7)	3.0
((14-5)/(9-6))	-5.0
((-6)+((7*2)/14))	

Problem B: The Election

I come from Camden town, which is famous for its diverse population. Number of people living in this town is N, each having a different amount as their bank balance. Every person has a social security number, which is a unique integer in the range of [**1,N**]. A person of social security number **i** has **x_i** bucks as their bank balance.

A mayor election is imminent in the town, so several candidates are registering every day to get voted (and hence elected). Candidate id is always unique for any particular candidate, and if a candidate C1 registers before another candidate C2, then candidate id of C1 will be smaller than candidate id of C2. The candidates are very honest, so they don't offer any promises in their manifesto that they can't really fulfil. So it has turned out that every candidate can only serve a group of people, people with social security numbers in the range of [**L,R**], where $\mathbf{1 \leq L \leq R \leq N}$. A candidate also has a dignity factor **m** (positive integer), which takes into effect as the way we will learn shortly. A candidate's promise can be defined as the tuple **(L,R,m,c)**, such that,

A candidate with candidate id **c** serves the people in **[L,R]** range of social security numbers by promising to increase their bank balance in the following way,

```
x_L' = x_L * m^0
x_(L+1)' = x_(L+1) * m^1
x_(L+2)' = x_(L+2) * m^2
.
.
.
x_R' = x_R * m^(R-L),
```

where `x_i'` is the new bank balance if this candidate is selected.

Each person gets to vote only once and to only one candidate. Of course any person would surely want to maximize their account balance, so time to time they wonder which candidate they should cast their vote for.

Constraints:

```
1 ≤ N ≤ 100000
1 ≤ Q ≤ 100000
2 ≤ m ≤ 1000000000
1 ≤ x_[i] ≤ 1000000
1 ≤ c ≤ Q
```

Input:

Input will start with an integer, number of test cases **T**, then **T** cases follow.

Each test case will start with an integer **N**, followed by **N** non-negative numbers in the next line.

In the next line there will be an integer **Q**, number of queries.

Each query will be in either format of the following,

Format 1: **1 L R m c** : a new candidate joins the election now with the tuple **(L,R,m,c)** as mentioned in the statement. It is guaranteed that no two candidate has same candidate id.

Format 2: **2 P** : Person with social security number **P** wonders which candidate they should vote for (among the candidates who have already joined the election by the time of this person's query)

Output:

For each test case, output the case number first. And then for each case, for each query of Format 2, output the id of the candidate that the person with security number **P** is going to vote for, followed by a space and the amount of his possible bank balance mod `1000000007`. If two or more candidates offer same amount of account balance to this person, then the candidate who registered first will get this vote. If a person has no gain in voting anybody, print `-1`.

Sample Input	Sample Output
1 6 1 1 1 1 1 1 11 1 2 5 3 1 1 1 4 2 2 2 2 2 3 2 4 2 5 2 6 1 3 6 5 3 2 4 2 5 2 6	Case 1: 2 2 2 4 1 9 1 27 -1 1 9 1 27 3 125

Problem C: Finding the Determinant

A tridiagonal matrix is a band matrix that has nonzero elements only on the main diagonal, the first diagonal below this, and the first diagonal above the main diagonal.

In this problem, consider a tridiagonal matrix, A_{nxn} , with ms on its main diagonal, `1`s in the adjacent diagonals below and above the main diagonal, and `0`s everywhere else. The sample matrices are as follows:

For `n=2`, $A = \begin{bmatrix} m & 1 \\ 1 & m \end{bmatrix}$, for `n=3`, $A = \begin{bmatrix} m & 1 & 0 \\ 1 & m & 1 \\ 0 & 1 & m \end{bmatrix}$, and $A_{n\times n} = \begin{bmatrix} m & 1 & 0 & 0 & \dots & 0 \\ 1 & m & 1 & 0 & \dots & 0 \\ 0 & 1 & m & 1 & \dots & 0 \\ 0 & 0 & 1 & m & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \dots & m \end{bmatrix}$ in general.

If the main diagonal elements are `2`s, then the determinants are 3 for n=2, and 4 for n=3.

Given `n` and `m`, determine the determinant of the given $n \times n$ tridiagonal matrix matrix with diagonal elements `m`, where `2≤n≤9, 2≤m≤10`, your task is to compute the determinant of that tridiagonal matrix.

Formula for finding the determinant

For a 2x2 matrix, the determinant is:

$$|A| = \begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$$

For a 3x3 matrix, the determinant is:

$$|A| = \begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} = a\begin{vmatrix} e & f \\ h & i \end{vmatrix} - b\begin{vmatrix} d & f \\ g & i \end{vmatrix} + c\begin{vmatrix} d & e \\ g & h \end{vmatrix}$$

The same sort of procedure can be used to find the determinant of a 4x4 matrix, the determinant of a 5x5 matrix and so on.

Input

There may be several data sets as multiple lines.

Each data set will consist of a single line of two integers, `n (size of the matrix) and m (elements on main diagonal) where 2≤n≤9, 2≤m≤10`.

The input will end with a line containing two `0`s.

Output

For each data set, print the determinant of the matrix with given input.

Sample Input	Sample Output
2 2 4 3 6 5 9 4 0 0	3 55 12649 151316

Problem D: Bakery Delivery Scheduler

The ABC Bakery offers one of the most unique bakeries in Myanmar, offering delicious snacks such as pastries, puffs, pains and breads and sweets, such as cakes, crème pies and cream-based desserts. It is also popular among people because they can give faster service for the customer orders. The reason they can give the customer satisfaction is that they use a better delivery scheduling for order services. The scheduling is as follows:

The customer orders the items with their home address. Then, the bakery estimates the address into some unit representing the distance. Suppose that the distances in the Queue are {`23, 89, 132, 42, 187`} and the shop/delivery is at distance `100.`

If the direction of delivery service is to go upward (it means going to the larger and nearest numbers) *first* and then goes down (it means going to the smaller and nearest numbers), the order in which the delivery service will go is `132, 187, 89, 42, 23.` In this case, the delivery went a total of distance `251` units.

If the direction of delivery service is to go downward *first* and then goes upward, the order in which the delivery service will go is `89, 42, 23, 132, 187`. In this case, the delivery went a total of distance `241` units.

Your task is to suggest the direction that the delivery should start in order to minimize the distance. In the above example, you would suggest the `downward` direction to go first since the total distance is small.

Input

There may be several test cases, `(1 ≤ t ≤ 20)`. Each test case consists of two: a series of request, and the starting location of the delivery. The series of `n (0<n<100)` requests in distance would be input one per line, ranging from `0` to `200`. The request may be ended up with `-999`. The starting location of the delivery would be from `0` to `200`.

Output

Your program should output the selected direction (`Upward_First/Downward_First`) and a total distance.

Example

See the samples for the exact format of output. There are two test cases (`t=2`) in this example. For t=1, the series of request is `23 to 187`. The delivery starts at distance `100`. For t=2, the series of request is `98 to 78`. The delivery is at distance `130`.

Sample Input	Sample Output
2 23 89 132 42 187 -999 **100** 98 183 37 122 130 65 78 -999 **130**	Downward_First 241 Upward_First 199

Problem E: Tree Pendant

You own a small gift shop which sells computer-aided designed and 3D-printed cute stuffs. For example, the following picture shows a tree pendant, which is quite popular among young ladies who like complex and mysterious-looking things.

Girls want their pendants to be unique, so you developed a program that generates random trees. However, some girls are really picky: they want some very specific modifications to make! During the (long) modifying process, they may even ask questions about some properties of the current tree, to make sure some statistics numbers look good.

Technically, the tree in the pendant can be described with a tree with n nodes, in graph terminology. In order to make it more colorful, each edge can be colored **gold** or **silver**, which can be denoted by 0 and 1, respectively. Each query is one of the followings:

- **COLOR u v k**

Color the unique simple path from u to v, with an alternating sequence starting with k. For example, if k=0, then the edges on the path will be colored gold, silver, gold, silver, ..., in this order.

- **REATTACH u v w**

Remove the edge u-v, flip the colors of all edges in the sub-tree rooted at node v, and then re-attach the sub-tree to node w (in other words, add another edge v-w). By "flip" we mean changing gold to silver, and changing silver to gold. The color of the new edge should be the same as the removed edge. Before the query, it's guaranteed that nodes u, v and w are all distinct, edge u-v exists but edge v-w doesn't. Moreover, w is not in the sub-tree rooted at node v after removing edge u-v, so after this query, the pendant is still a valid tree (i.e. connected and loop-less).

- **PATH-STATS u v**

Print the number of gold and silver edges along the unique simple path from u to v.

- **TREE-STATS u v**

Print the number of gold and silver edges in sub-tree rooted at node u, if we temporarily regard node v as the root of the tree.

Input

There will be at most **10** test cases. Each case begins with two integers **n** and **q (1 ≤ n ≤ 100000, 1 ≤ q ≤ 100000)**. The following **n-1** lines describe the initial tree. Each line contains **3** integers **u**, **v** and **c**, that means there is an edge between node **u** and **v**, with color **c (1 ≤ u, v ≤ n, 0 ≤ c ≤ 1)**. Each of the following **q** lines is a query. The first integer in the line is the type of query: **1** means **COLOR**, **2** means **REATTACH**, **3** means **PATH-STATS**, **4** means **TREE-STATS**. For all queries, **1 ≤ u, v, w ≤ n, 0 ≤ k ≤ 1**, and parameters obey all restrictions stated above. The total number of queries in all test cases is at most **100000**.

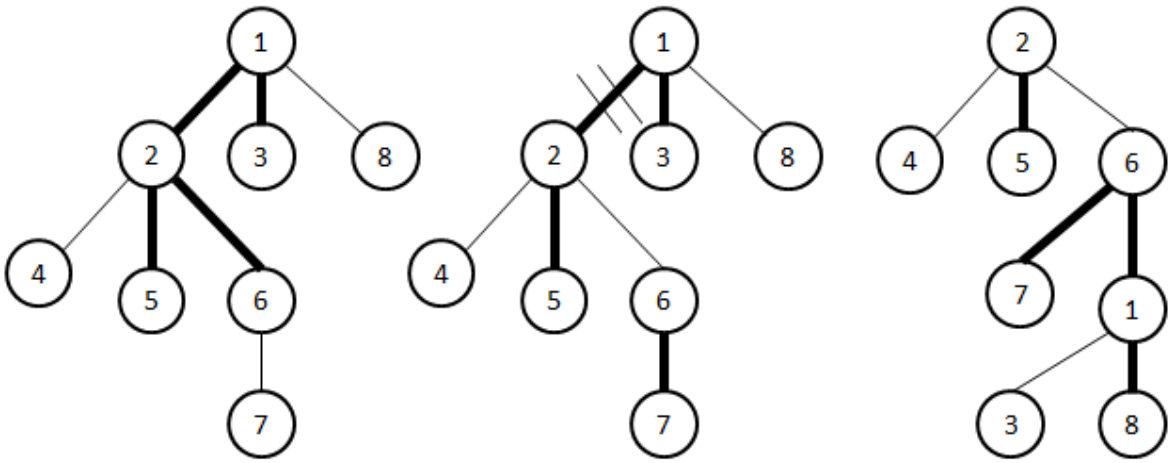
Output

For each test case, print the case number in the first line. Then print your answer in a separate line for each of **PATH-STATS** or **TREE-STATS** query.

Sample Input	Sample Output
8 2 1 2 0 1 3 0 1 8 1 2 4 1 2 5 0 2 6 0 6 7 1 3 7 8 4 1 1 8 4 1 2 0 1 3 0 1 8 1 2 4 1 2 5 0 2 6 0 6 7 1 1 7 1 0 3 6 3 2 2 1 6 4 1 3	Case 1: 2 2 4 3 Case 2: 2 1 4 2

Explanation

The evolution of the tree is as follows (thick edge is gold):



Case 1, Query 1: (left picture) path 7-6-2-1-8 has 2 gold and 2 silver edges.

Case 1, Query 2: (left picture) the whole tree has 4 gold and 3 silver edges.

Case 2, Query 2 (middle picture): path 6-2-1-3 has 2 gold and 1 silver edges.

Case 2, Query 4 (right picture): if we regard 3 as the root, sub-tree 1 has 4 gold and 2 silver edges.

Problem F: Chocolates

The Association of Chocolate Industry (ACI) is preparing to launch a new product. Its idea is old with a novel twist: it simply sells boxes of chocolates. But since people are what they consume and everyone wants to be unique these days, the ACI wants every chocolate box to be unique, in the sense that no two boxes should contain the same composition of chocolate types.

The ACI is only able to make a small number n of different types of chocolate, but while limited in imagination, it is virtually limitless in resources, so it is able to produce as many as it wants of each type of chocolate. Furthermore, the chocolate types have different weights (though some may weigh the same), and in order to simplify pricing matters, the ACI wants all chocolate boxes to have the same total weight.

With these restrictions, the ACI will only be able to make a limited number of boxes. For instance, if there are three types of chocolate, weighing 5, 5 and 10 grams respectively, 4 different boxes (i.e., for four people) can be made with total weight 10 grams (using either two of type1, or two of type2, or one of type3, or one each of types1 and 2). The ACI would like to be able to make at least one box for everyone in the cosmos. So, given queries in the form of the number of people P in the cosmos, your job is to find the smallest possible total weight w such that P different boxes containing exactly w grams of chocolates can be made.

Input

The input consists of several test case, t ($1 \le t \le 20$). Each test case consists of four lines. The first line contains an integer $1 \le n \le 5$, the number of chocolate types. The next line contains n integers $w_1, \ldots, w_n$, where $1 \le w_i \le 10$ is the weight (in grams) of the i^{th} chocolate type. The third line contains an integer $1 \le q \le 10$, the number of queries. The last line of each test case contains q integers $P_1, \ldots, P_q$, where $1 \le P_j \le 10^{15}$ is the number of people.

Output

For each test case, t, write "Set_t" in a separate line, followed by q lines giving, for each query P_j, the minimal possible positive weight W_j (in grams) of a chocolate box. If there is no weight W_j such that at least P_j chocolate boxes can be made, print "NONE" for that query.

Sample Input	Sample Output
4 3 5 5 10 1 4 4 4 6 2 8 2 200 550 1 10 1 50 4 5 5 7 9 3 10 20 30	Case 1: 10 Case 2: 52 76 Case 3: NONE Case 4: 30 44 54

Explanation

There are four test cases and the numbers in the first case represents as follows:

3 - represents the number of chocolate types.
5 5 10 - represents the weights of each type.
1 - means the number of queries.
4 - means the number of people in above query.

The result , **10**, in the first test case is the smallest possible weight in each box for four people.

Problem G: Vampire Numbers

In mathematics, a vampire number v is a composite natural number v=AB, with an even number of digits n, that can be factored into two integers A and B each with n/2 digits and not both with trailing zeros, where v contains precisely all the digits from A and from B, in any order, counting multiplicity, A and B are called "fangs".

For example: 1260 is a vampire number, with 21 and 60 as fangs, since 21 × 60 = 1260. However, 126000 (which can be expressed as 21 × 6000 or 210 × 600) is not, as 21 and 6000 do not have the correct length, and both 210 and 600 have trailing zeroes. Similarly, 1023 (which can be expressed as 31 × 33) is not, as although 1023 contains all the digits of 31 and 33, the list of digits of the factors does not coincide with the list of digits of the original number.

Examples of vampire numbers (in ascending order) are:

```
1260 = 21 x 60
1827 = 21 x 87
102510 = 201 x 510
105264 = 204 x 516
```

Given the lower and upper bounds of an interval, find the vampire numbers in the given interval, output these numbers, one per line if exists, otherwise output "**NONE**".

Input

The first line of input will begin with an integer t $(1 \le t \le 1000)$ on its own line, i.e. the number of data sets.
The next t lines will consist of two integers, L and U $(10 \le L \le U \le 10^8)$.

Output

For each data set, print the vampire numbers (in ascending order) between L and U, inclusive, in the format:
v = A x B where $A \le B$.

Sample Input	Sample Output
3 1000 2000 10 188 450000 460000	Case 1: 1260=21x60 1395=15x93 1435=35x41 1530=30x51 1827=21x87 Case 2: NONE Case 3: 456840=540x846 457600=650x704 458640=546x840

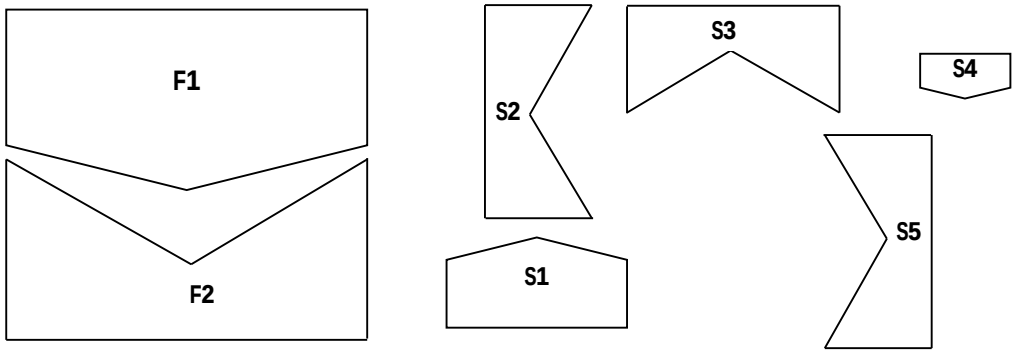
Problem H: Like Father Like Son

In a city from another world, citizens are polygons. Each family's shape is different from others. But the children are in the same shape as their parents although they have different little sizes than their parents. Therefore they can be remembered that who are their parents. The parents encourage their Kid's to get active on Winter School Holidays. They usually enjoy a winter of Fun at adventure playground, a free, exciting, creative, adult restricted play space for children to play freely, in their own way, in their own time. There is no limit to the play opportunities available, freely chosen and imagined by children. They also very happy to play on that playground and play none stop on a lot activity.

A problem will begin at the playground's closed time. Fathers waiting outside the playground cannot remember their children. Because their children's shape are in 90, 180 and 270 degree rotation according to many activity games. Sometimes they are in reflected shape. At that time, the playground security has to check not to miss the fathers and their own sons. There are three main steps:

1. Firstly, the children and their fathers' center points are needed to translate the security door of coordinate (0,0).
2. Secondly, their sizes are scaled at this coordinate origin (0,0) to get their fathers' size.
3. Finally, they are rotated back in 90, 180 or 270 degree or reflected back using
   $x' = x * \cos\theta - y * \sin\theta$ and
   $y' = x * \sin\theta + y * \cos\theta$ to get the normal position like their fathers.
   For X-axis reflection, $x' = x$ and $y' = -y$.
   For Y-axis reflection, $x' = -x$ and $y' = y$.
   In this way, the parents can take their Like Father Like Sons on the way of home back.

As shown in figure, the sons of father F1 are S1 (a half size and reflection) and S4 (a quarter size). The sons of father F2 are S2 (a half size and rotate clockwise 90 degree), S3 (a half size and reflection) and S5 (a half size and rotate counter clockwise 90 degree).



Input

The first line contains the two-separated input which are a character '**F**', the symbol of father and **T**, the number of fathers. The second line contains the two-separated input which are string (father name F) and integer (the number of vertices of the polygon **V)**. This is followed by V lines, each containing the two space separated real numbers **x** and **y** coordinates for a polygon vertex.
After completing the number of polygon vertices, the next line contains the two-separated input which are a character '**S**', the symbol of son and **T**, the number of sons. Then the next line contains the son name **S** and the number of vertices of the polygon **V** which are followed by V lines, each containing the two space separated real numbers **x** and **y** coordinates for a polygon vertex.

Output.

For each Father name, print the Father name and their own sons name in one line of separated strings.

Constraints

- $1 \leq F \leq 10^5$
- $1 \leq S \leq 10^5$
- $3 \leq n \leq 30$
- -1.79769313486231570E+308≤ x, y≤ +1.79769313486231570E+308 in Java and 1.175494351 E – 38≤ x, y≤3.402823466 E + 38 in C++
- Father's size must be greater than or equal to their son's one.
- The son must be in a reflection of X-axis or Y-axis or in a rotation at 90, 180 or 270 degree in clockwise or counter-clockwise direction or in equal shape like his father.

Sample Input	Sample output
F 2 **F1 5** 4 4 2 6 2 8 6 8 6 6 F2 5 10 1 10 3 12 2 14 3 14 1 **S 5** **S1 5** 8 6 8 7 9 8 10 7 10 6 **S2 5** 13 7 13 9 14 9 13.5 8 14 7 **S3 5** 7 10 7 11 9 11 8 10.5 9 10 **S4 5** 1 1.5 1 2 2 2 2 1.5 1.5 1 **S5 5** 6 2 6.5 3 6 4 7 4 7 2	F1 S1 S4 F2 S2 S3 S5

Problem I: Clustering

The analyst wants to analyze the data set that contains numeric value. The data set is represented as 2D matrix of `0` and `1`. The task is to count the number of cluster that formed by value `1` in the data set.

For example, in the following 2D matrix representation of data sets shows the number of such clusters.

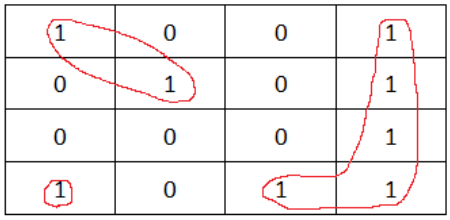


Figure 1: total of three clusters

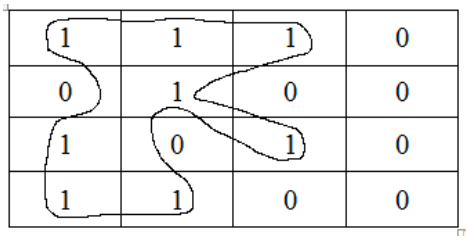


Figure 2: total of one cluster

Your job is to write the program that find total number of clusters formed by elements with value `1` in a given data set.

Input

There may be several test cases, `(1 ≤ t ≤ 20)`. Each test case consists of two: row and column of 2D matrix, and data in matrix.

The number of rows, `n` `(1 ≤ n ≤ 100)`, and the number columns, `m` `(1 ≤ n ≤ 100)`, are input to the system as one line separated by a blank.

Then `n` data lines would contain `m` number of `0 (or) 1` separated by a blank.

Output

The output line consists of the number of clusters in the data set.

Sample Input	Sample Output
2 4 4 1 0 0 1 0 1 0 1 0 0 0 1 1 0 1 1 3 5 1 0 0 1 0 1 0 0 1 0 1 1 1 0 0	3 1

Problem J: Green Frog and Ordering

Little Green Frog has a list of numbers. On this list, she can increase any number by one. When a number is increased, it moves to the front of the list. All the numbers those were previously to the left of the increased one, moves by one index to the right. In other words, if the i^{th} number was increased, then,

1. The i^{th} number moves to position 0,
2. Each of the numbers previously at 0th to $i\text{-}1^{th}$ position moves to the right by 1 position of its initial position.

Green Frog wants to sort the given list in strictly increasing order by doing minimum number of operations. Please print the minimum number of operations required to sort the list.

Input:

The first line of the input has a single positive integer number, `T, T<= 100`, which is the number of test cases. Each of the test cases has two lines.
The first line has a positive integer `N, N <= 100000(10^5)` and in the second line, there are `N` positive integer numbers denoting the initial list Green Frog has. None of these numbers are greater than `10^9`. Numbers on the list are separated by single spaces.

It's guaranteed that there are no more than `10^6` numbers in the test file.

Output:
For each test case, print a single integer number denoting the minimum cost to sort the given list in ascending order.

Sample Input	Sample Output
3 5 1 2 3 4 5 5 2 1 3 4 5 3 3 1 4	Case 1: 0 Case 2: 5 Case 3: 1

Note:
In case 1, the given list is already in increasing order. So no operations are necessary to sort it.
In case 3, we increase the 2nd number by 1, so it become 1+1 = 2 and then moves to the front. So the resulting array after one operation is (2, 3, 4), which is sorted in increasing order. So the answer is 1.

Problem K: One Time Pad

Mr. Smith wants to send a secret message to Mrs. Smith. But, he worries that his sending message can be intercepted by an intruder. So, he decides to encrypt the message using one time pad. The one time pad (OTP) is an encryption technique that cannot be cracked, but requires the use of a one-time pre-shared key the same size as the message being sent. In this technique, a plaintext is paired with a random secret key. Then, each character of the plaintext is encrypted by cyclically shifting to the right by its corresponding 'key' times from the pad. The encrypted message should be the characters between ‘a’ and ‘z’ inclusive. If the character is shifted past ‘z’, it starts back at ‘a’ and continues shifting. Suppose that the message is “ucsy” and the keys are {2, 2, 3, 4}; and then, the message is encrypted as “wevc” by cyclically shifting to the right. Any characters except ‘a’ to ‘z’ in the plain text would be encrypted as ‘?’.

Input

The first line of the input contains a single integer `t, (1<=t<=10)` that indicates the number of test cases. Each test case contains `2` inputs, one per line. The first input is the message to be encrypted. The message should be one sentence/phrase in a single line. This would include lowercase characters ('a' - 'z') and other symbols (,.?). The second input is the series of keys to be used in encryption of the input message. The keys are integers separated by a blank. The number of keys and the length of message would be equal.

Output

The output should display the encrypted message for each input.

Example

See the samples for the exact format of output.

Sample Input	Sample Output
3 prime minister 1 2 3 4 4 5 5 5 6 7 1 2 3 3 contest 2 2 2 3 3 3 4 yes, i will do. 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15	qtlqi?rntptvhu eqpwhvx zgv??o?ervw?qc?

Problem L: Passwords for Sweethearts

Acton and Carlie are young lovers. They love and trust too much to each other and they share the same password for the social networks login, such as facebook, twitter, etc. However they use some policies as follows to create the passwords:

- Any password must contain the combination of two words "atn" for Acton and "cl" for Carlie.
- The password must be in lowercase.
- The password must be interleaved in a way that maintains the left to right ordering of the characters from each word.

According to the above policies, they start use "`caltn`" as the very first password. But, the two lovers don't know how many more passwords they can create later. Actually, there are `9` more valid passwords, `"actnl", "clatn", "acltn", "actln", "atcln", "atncl", "catnl", "catln", "atcnl"`.

So, you are supposed to help such lovers by writing a program in creating their interleaved passwords. Your program finds the distinct interleaved passwords that can be constructed by combining from the two words.

In the following example, there are `6` distinct interleaved passwords, `baba, aabb, abba, abab, baab, bbaa`, that you can get from "`aa`" and "`bb`".

However, there is only one distinct interleaved password, "`aaaaaaa`", from "`aaa`" and "`aaaa`".

Input

The first line of the input contains a single integer `t, (1<=t<=10)` that indicates the number of test cases. Each test case contains `2` words in a single line, separated by a blank, each containing only lowercase `'a'` to `'z'` characters. The length of each word is between `1` and `5`, inclusive.

Output

For each test case, output the valid passwords list in dictionary order (ascending order) followed by the total number of distinct interleaved passwords. See the samples for the exact output format.

Sample Input	Sample Output
3 atn cl kp py aaa aaaa	acltn actln actnl atcln atcnl atncl caltn catln catnl clatn 10 kppy kpyp pkpy pkyp pykp 5 aaaaaaa 1