A Back to the Past

Input: Standard Input
Output: Standard Output

Dr. James, the only PhD holder of our country in Competitive Programming (CP), has recently joined the Department of Competitive Programming in St. John Snow University as a professor. He has passed a lot of memorable times in this institute as a student. He was a hero among the seniors and juniors. He also loved to spend good times with others, taking voluntary classes, helping students to learn different algorithms and data structures.

"Alas! The good old time. I wish if I could go back to the past" – Professor Dr. James mumbles. Though he is now a teacher in the institute, he cannot maintain the previous interactions and activities with students. Dr. James wants to be the man like before but some senior professors does not like this attitude. They do not appreciate the friendly interaction between teacher and student.

Now Professor Dr. James dreams about going back to the past. As he knows that this is not possible, he is just giving himself consolation seeing the calendar.

Suddenly Dr. James recalled today's date (31/10/2015) as the preliminary contest of ACM ICPC Asia Regional Dhaka site 2015 is going to start on this date. As he is one of the problem setters and judge, he wants to play a (hard) game with the contestants. Name of the game is "Back to the Past".

The rule of the game is simple. Professor Dr. James has chosen a random past date. You have to print the date and the day of the week of that date. The chosen date by Dr. James is **29/05/2013** (nobody knows the reason). See the output section for more information.

Input

There is no input in this problem.

Output

Output the date and the name of the week day.

Sample Input	Output for Sample Input
	October 30, 2015 Friday

The sample output will give wrong answer as this is a sample to show the output format.

Hint: Code snippet in C/C++ to print the above sample output

```
#include<stdio.h>
int main()
{
     printf("October 30, 2015 Friday\n");
     return 0;
}
```

You just need to replace October 30, 2015 Friday with the correct answer.

B Search the Khoj

Input: Standard Input
Output: Standard Output

Jalil is a five year old wonder boy. He has a very strange memory power. He can memorize any number in his brain with at most one digit error. If you ask him to tell first 1000 digits of PI (3.141592…), he will go through it without any hesitation. But he could have made a mistake in at most one digit, whether you ask him to tell first 100 digits or 5 digits. Suppose you asked him to tell first 4 digits of PI after decimal place. Probable answer from him could be 1415, 1416, 0415, 1475 etc. but not 0516, 1517 etc.

Due to ACM ICPC Preliminary contest, today is a school holiday for Jalil but not for his mommy. He is feeling bored and thought about calling mommy to come home early. But his mommy has a new mobile phone with new number. Though his mommy told him the number but he is afraid he might have forgotten one of the digits of the mobile number. He took his dad's mobile and searched for his mommy's number in the contact list, but could not succeed. Jalil is cent percent sure that one of the numbers in the contact list of his dad's mobile is his mommy's number. So he took a backup of his dad's contact list in his computer and started matching the number he can remember with all the numbers in the contact list.

As Jalil know that he might have mistaken at most one digit in his mommy's mobile number, he is preparing a list of mobile numbers from his dad's contact list such that each of the numbers is only at most one digit different from the number he remembers. As Jalil does not know programming, help him to prepare the list.

Input

First line of the input contain a positive integer **T (T≤10)**, the number of test cases. First line of each test case contains a positive integer number **N (≤1000)**, number of mobile number in the dad's contact list. Each of the next **N** lines will contain a mobile number from dad's contact list. Next line will contain mommy's phone number which Jalil can recall. A mobile number is a non-empty string of numeric characters (**0** to **9**, leading zero allowed) of length at most **11**. Every mobile number (contact list and mommy's number) given in a single test case will be of same length. Note that, **0123** and **00123** are different mobile numbers.

Output

For each test case, print the test case number in a single line followed by the list of probable number of mommy. Print the phone numbers according to the order in the input.

Sample Input	Output for Sample Input
2 3 0123456 0012345 0123457 0123458 2 123 124 123	Case 1: 0123456 0123457 Case 2: 123 124

C Moroccan Wooden Box

Input: Standard Input
Output: Standard Output

Three famous mathematicians Erich Friedman, David Cantrell and Craig Clapp are visiting Morocco to attend the International Computational Programming Conference. As they only plan to visit the conference to enjoy what other people say, they have left all their computers, calculation details back home. But out in the street see many Moroccan Wooden boxes and they plan order two boxes which will be used to keep certain number of things of fixed shape. For simplicity the following things are assumed by them:

1. The two boxes will have square shapes (As shown in the figure on the left).

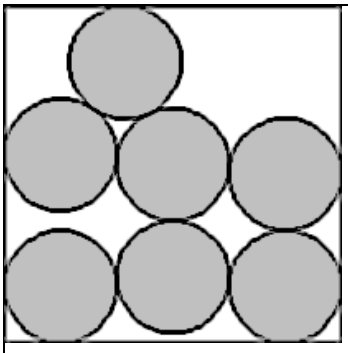
2. The boxes will be used to keep balls of fixed radius.

3. Exactly seven or eight sphere shaped balls of radius **r** will be placed in the box.

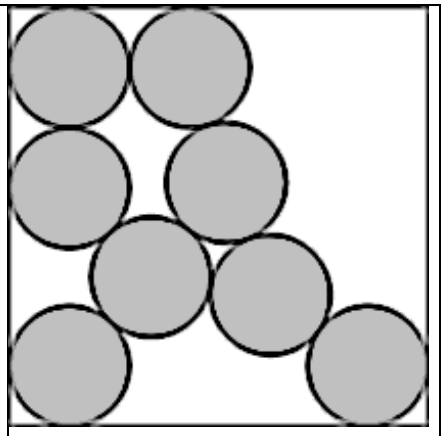
4. The boxes should be as large as possible but the things kept inside must remain rigid (should not be able to move in any direction) when the box is closed, regardless of the orientation of the box.

5. The thickness of the box walls should be considered **w** units.

Now given the radius r of all the balls and the constraints given above, you will have to find the maximum possible volume of two boxes that contain 7 and 8 spheres respectively. As Erich, David and Craig are experts in this field so they also provide you two dimensional sketches of the positioning of the balls that should make the volume maximum. The sketches are shown below



Sketch 1: Top view of the cross section (At height r from the bottom) of a box containing seven rigid balls.



Sketch 2: Top view of the cross section (At height r from the bottom) of a box containing eight rigid balls.

The figures above are accurate but not accurate enough to find the volume of the boxes exactly from the figures only. Some important things about the figures above are (a) When two balls appear to touch each other or the wall, they actually do touch (Otherwise the balls will not remain rigid) (b) When three circles appear collinear in naked eye, you cannot assume that they are actually collinear (c) If the circle appears to have equal radius in the cross section, they actually have equal radius.

Input

The input file contains around **1000** lines of inputs. Each line contains two floating-point numbers **r** and **w (0.01 ≤ r, w< 500)**. Here r is the radius of all the balls and **w** is the thickness of the box-wall. Input is terminated with end of file.

Output

For each line of input produce one line of output. This line contains two floating-point numbers V_7 and V_8 with four digits after the decimal point. This V_7 denotes the maximum possible volume of the box that contains **7** balls and V_8 denotes the maximum possible volume of the box that contains **8** balls. Absolute error of less than 10^{-4} or relative error of 10^{-9} will be ignored.

Sample Input	Output for Sample Input
0.01 0.01 0.02 0.01	0.0003 0.0003 0.0012 0.0015

D XOR Subset

Input: Standard Input
Output: Standard Output

Fermat's little theorem states that if ***p*** is a prime number, then for any integer ***a***, the number ($a^p - a$) is an integer multiple of ***p***. In the notation of modular arithmetic, this is expressed as

$$a^p \equiv a \pmod{p}.$$

For example, if $a = 2$ and $p = 7$, $2^7 = 128$, and $128 - 2 = 7 \times 18$ is an integer multiple of 7. We can also write 128 % 7 = 2, here % is the modulo operator used in C/C++ or Java.

If a is not divisible by p, Fermat's little theorem is equivalent to the statement that $a^{p-1} - 1$ is an integer multiple of p, or in symbols

$$a^{p-1} \equiv 1 \pmod{p}.$$

For example, if $a = 2$ and $p = 7$ then $2^6 = 64$ and $64 - 1 = 63$ is a multiple of 7. We can also write 64 % 7 = 1.

You are given a set S which contains 1 to N. You want to find two subsets of S, X and Y such that the following conditions are met:

1. $X \cap Y = \varnothing$
2. Let bitwise XOR of every element of X equals U and Y equals V. U must be less than or equal to V.

You want to find out number of ways you can choose such subset X and Y. Two ways (X1, Y1) and (X2, Y2) will be equal if X1 equals X2 and Y1 equals Y2 or X1 equals Y2 and Y1 equals X2.

For example is S = {1, 2}, the ways are:

1. $X = \varnothing$, $Y = \varnothing$. [U = 0, V = 0]
2. $X = \varnothing$, $Y = \{1\}$. [U = 0, V = 1]
3. $X = \varnothing$, $Y = \{1,2\}$. [U = 0, V = 1 ^ 2 = 3, (^ means bitwise XOR in C/C++/Java)]
4. $X = \varnothing$, $Y = \{2\}$. [U = 0, V = 2]
5. $X = \{1\}$, $Y = \{2\}$. [U = 1, V = 2]

Now, given **N**, you need to find the number of ways you can choose two subsets of **S** such that the 2 conditions meet, modulo **1000000007 ($10^9 + 7$)**.

Input

First line contains **T (T<=100)**, the number of test cases. Each of the next **T** lines each contains an integer **N (0<=N<10^{10000})**.

Output

For each case print one line, "**Case C: W**", where **C** is the case number, and **W** is the required answer for that case.

Sample Input	Output for Sample Input
2 2 3	Case 1: 5 Case 2: 14

E Emoticons

Input: Standard Input
Output: Standard Output

Nowadays emoticon has become an art. People are no longer limited to simple ones like :-), :-(, :-P etc. They use >:O, ~_~, =^_^= and so on. Recently I came across ^_^ and it looks kind of cute to me. Given a string S consisting of only _'s and ^'s, I was wondering what is the maximum number of disjoint subsequences of "^_^" (quote for beauty) in the string S.

For example, if S = "^^__^^" then the answer is 2. However, for S = "_^^" the answer is 0.

Input

Input starts with a positive integer **T (<= 5,000)**, denoting the number of test cases. Hence follows **T** test cases. Each case consists of a single string made of only ^ and _. The length of the strings would be at most **100,000** and the sum of lengths of the strings will be **2,100,000** at most.

Output

For each test case, print the case number followed by the answer.

Sample Input	Output for Sample Input
5 _^^_^^_ ^_^_^ ^_^^ _____ ^^_^^ ^_^^_^	Case 1: 1 Case 2: 1 Case 3: 0 Case 4: 2 Case 5: 2

Hint:

- S[1...n] means S is a string of length n and it is 1-indexed.
- S_i means i'th character of S.
- A string S[1...n] is a subsequence of another string T[1...m], if we can find: (t_1, t_2, … t_n) such that, S[i] = T[t_i] for 1 <= i <= n and 1 <= t_1 < t_2 < … < t_n <= m. For example, "abc" is a subsequence of "aabbcc" but not of "bca".
- Two subsequences are disjoint if same character (position matters) is not used in both of the subsequences. For example, let S = "abca". "ab" and "ca" are two disjoint subsequences of S. However, if S = "abc" then "ab" and "ac" are not disjoint subsequences. In both of these examples the subsequences are unique. However, for S = "aabb" let's form two subsequences S_1S_3 and S_2S_4 (both are "ab"), both of these are disjoint. But if we have chosen S_1S_3 and S_1S_4 then they would not be disjoint.

F Brain Fry

Input: Standard Input
Output: Standard Output

A programming contest is happening at Rightshift University of Science and Technology in Rightshift city. After the mock contest, the judges become hungry and decide to try local Brain Fry (Mogoj Vuna) with Maskalai roti (bread). But as the brain-fry is very popular dish, it is not guaranteed that if judges go to a restaurant, they will find it there. They learned from the volunteers that there are N restaurants in Rightshift city numbered **1** to **N**. For the i'th restaurant, the probability to find brain fry is P_i **(1<=i<=N)**. But even if they don't find brain fry at any specific time at a restaurant, they might find it at another time, because the restaurants might refill the brain fry. The probability to find brain fry at restaurant **i** will remain the same ($\mathbf{P_i}$).

The judges are at the university which is denoted by place **0**. The Rightshift city can be modeled by an undirected weighted graph where each edge is represented by three numbers **u, v** and **w**. Here **u** and **v (0<=u,v<=N)** are two places (**0** is the university and **1** to **N** are restaurants) and w is the required time needed to travel between **u** and **v**. For excessive flyways, place **u** and **v** don't necessarily needs to be physically adjacent. There can be at most one road between any two pair of places.

Now, the judges are planning to do something crazy. They decided to spent **T** times to search for brain fry. They have following strategy:

1. Starting from university (place 0), they will select a random neighbour restaurant, go to that restaurant and follow the step 2. If there is no neighbour restaurant or **T** time has been elapsed, then they will just quit.
2. At arriving restaurant **i**, they will check for brain fry here. If they find it (probability of finding it is, $\mathbf{P_i}$) they will eat it instantly (they are really hungry) and will go back to the university (place 0) using shortest distance and stop their search for brain fry.
3. If they don't find it at restaurant **i** and **T** time has been elapsed, they will go back to the university (place 0) using shortest distance. And they will compensate it in the next day's contest by making fry out of contestants' brain.
4. Otherwise they will select a random neighbour restaurant again and go to that restaurant and start from step 2 again. If there is no neighbor restaurant from current restaurant, then they will go back to the university (place 0) using shortest distance and start from step 1.

Note that, you want to use at most **T** time but there are some situations where judges have to use more than T time according to the rules above.

What is the probability that they will get to eat brain fry? What is the expected time they will take to return to the university?

Input

First line of the input is **TC(<=200)**, then **TC** test cases follows in next **TC** lines. First line of each case contains three integer **N (1 <= N <= 250)**, **M (0 <= M <= Min(N(N+1)/2, 20000))**, **T (1 <= T <= 100)**. Next line contains **N** non-negative real numbers, $\mathbf{i^{th}}$ of them is $\mathbf{P_i}$ **(0<=P_i<=1)** means that the probability of finding brain fry in $\mathbf{i^{th}}$ restaurant. Each of the next **M** lines contains three integer numbers **u, v** and **w (0<=u,v<=N** and **1<=w<=10)**. They represent there is a road between place **u** and **v** which takes **w** time to travel. No **u, v** pair will be given more than once.

Output

For each test case print a line in "**Case I: P E**" format where **I** is case number, **P** is the probability that they will get to eat brain fry and **E** is the expected time they will take to return to the university. Errors less than **1e-5** will be ignored.

Sample Input	Output for Sample Input
1 1 1 1 0.5 0 1 1	Case 1: 0.50000 2.00000

G Geek Power Inc.

Input: Standard Input
Output: Standard Output

Geek Power (GP) Inc. has recently invented a new kind of power source. They are just like normal power sources, but with one added limitation. You may ask at this point, who would use something new that has more limitations than the existing ones? Well, then you do not know geeks very well.
Each of these power sources has a power output rating with it and it can even be used with other power sources of different rating made by the GP only. But here is the limitation – if power sources of various ratings are mixed together, all of them start working at the lowest rating among them. For example, if you add three power sources with rating 3, 5, and 7, then each of them work as a power source of rating 3. Thus, the total power output from this group will be 3 x 3 = 9. However, if you only take 5 and 7 in the group, then each of them work as a power source of rating 5, and thus the total power output is 2 x 5 = 10, which is better than the previous group.
In this problem, you will be given a set of power sources made by GP. You have to form a group that produces the highest power output. You can take any number of power sources with any rating from the given list of power sources.

Input

First line of input is T (at most 200), the number of test cases. For each test case, you will be given an integer n (1 <= n <= 50), the number of groups of power sources with same power rating, and then n pair of integers (ki, pi) follows. Here, ki is the number of power sources with a rating of pi. You can assume that, 1 <= ki <= 100000 and 1 <= pi <= 1000.

Output

For each test case, print the case number starting with 1 and then the maximum power output that you can achieve. Read sample input output section for details.

Sample Input

```
3
3
1 3
2 5
1 9
4
2 5
2 1
2 1
2 5
5
2 1
4 1
2 1
3 3
6 10
```

Output for Sample Input

```
Case 1: 15
Case 2: 20
Case 3: 60
```

H Marbles in Jars

Input: Standard Input
Output: Standard Output

There are N jars. Each of the jars are labeled from 1 to N. Each jar contains marbles. The quantity of marbles in the jars is M1, M2, M3 … Mn, where M1 is the number of marbles in the first jar and so on. All the jars contain marbles that weighs 1 gram, except one jar that contains marbles weighing 1.1 grams. Let's call this jar **"The Fat Jar".**

You have a weighing machine, but you are allowed to use it **exactly once**. You need to find out the fat jar.

One neat way to find the fat jar is, (assuming we have enough marbles in each jar) take 1 marble from the 1st jar, 2 from 2nd jar, 3 from 3rd jar and so on. Let's say we have 4 jars in total. So ideally, the marbles should weigh 10grams collectively. Suppose, it weighs 10.3 grams. What does that tell us? The third jar has to be the fat jar. Because the extra 0.3 grams must have came from the three marbles that we took from the third jar.

Interestingly, there are several other ways to find out the fat jar. We call each way a **"Strategy".** Formally, a strategy is an array of N positive numbers: X1, X2, X3 … Xn, such that we take X1 marbles from the first jar, X2 marbles from the second jar and so on. We weigh X1 + X2 + X3 + … + Xn marbles in the weighing machine and try to find out the fat jar. A strategy is called **"Winning"** when it is always possible to find the fat jar following that strategy.

Given **N** jars with the number of marbles in each jar, how many different winning strategies are there? Two strategies are different if there exists an index **i**, for which **Xi** is different between those strategies.

Input

First line contains the **T (0<T<=100)**, the number of test cases. Each case starts with **N (1<=N<=100)**. Next line contains **N** integers **M1, M2, M3 … Mn**. (For all **i, 1<=i<=N, 0<=Mi<=100**).

Output

For each case, print one line, "**Case C: A**" (without the quotes). Here **C** is the case number and **A** is the answer that case. As the answer can be pretty huge, print the answer modulo **1,000,000,007**.

Sample Input	Output for Sample Input
3 3 1 2 3 3 1 2 2 2 2 2	Case 1: 1 Case 2: 0 Case 3: 2

Explanation:

The winning strategies in case 3 are {1, 2} and {2, 1}.

I

Jumping Frogs II

Input: Standard Input
Output: Standard Output

At time 0, F frogs are sitting on a straight line. All the positions of the frogs are non-negative integer numbers. Every second, all the frogs jump. Each of the frogs has its own velocity, i.e., every second the $\mathbf{i^{th}}$ frog jumps $\mathbf{V_i}$ units. Every frog jumps to its right.
The line is divided into **N + 1** contiguous segment. The left end of the first segment is always 0 and the right end of the $\mathbf{(N+1)^{th}}$ segment is $\mathbf{10^9}$. The segments are denoted by a sequence of **N** positive integers, the right end point of first **N** segments. Every segment except the first one starts from the first point after the right endpoint of the last segment.

For example, if N = 1 and the sequence has 1 integer number 10, then there are two segments, one is from 0 to 10 and another is from 11 to 10^9, both inclusive.

You are given the initial positions of all the **F** frogs and a sequence of positive integers describing the segments. Find the minimum time it will take all the frogs to reach a single segment. A frog is said to be on a segment if and only if it's sitting on some points inside the segment (including the endpoints). Please note that a frog is not said to be inside a segment when it's jumping.

Input

Input starts with a single positive integer, **1 <= T <= 10**, on a single line, denoting the number of test cases. Each of the following T test cases has the following 5 lines,

1. Blank line. To separate cases.
2. Two non-negative positive integers **1 <= F <= 1000, 1 <= N <=100,000**.
3. F non negative integers, where the $\mathbf{i^{th}}$ integer represents the position of the $\mathbf{i^{th}}$ frog.
4. F non negative integers, where the $\mathbf{i^{th}}$ integer represents the velocity of the $\mathbf{i^{th}}$ frog.
5. A sequence of **N** positive integers describing the segments.

Note that, all the numbers in the input are greater than **0** and less than $\mathbf{10^9}$ where a limit is not specified.

Output

For each case, print the minimum time it takes all the frogs to reach a single segment. If it's impossible for all the frogs to be on a single segment, print **-1**. For every case print the output on a single line.

Sample Input

```
2

1 1
10
10000
1000000

2 1
1 200
199 100
100
```

Output for Sample Input

```
Case 1: 0
Case 2: 1
```