13^{th} Iran Internet Programming Contest

Sponsored by University of Tehran - College of Engineering
University of Tehran ACM Student Chapter

Hosted by ShareCode

A – Iranian Community of Prize Crafting

Description

Iranian Community of Prize Crafting (ICPC) decided to provide some snacks and prizes for the most secure laboratories across the country. They prepared a list of n items (hereafter, called item list) they need! One of the ICPC members went to a great marketing store and bought m item(s). Some of these items might not be in the item list.
Write a program which calculates which items from the item list are still needed.

Input

The input contains several test cases.
In the first line of input comes T $(0 < T \leq 128)$, the number of test cases.
Each test case begins with two space-separated numbers n and m. The next two rows of the input contain n and m space-separated integers respectively. Each of these integers represents the code of each item. m and n are not greater than 1000 and the code of items are also lower than a billion.
To simplify, we only need one sample for each item. Therefore, the item codes in each row are unique.

Output

For each test case, you should print a line containing a sorted list of items (ascending order) that are still needed. The numbers should be separated by ",".

Sample Input

```
1
3 2
1 6 2
2 5
```

Sample Output

```
1,6
```

B – **EdoceRahs Triangle**

Description

You probably know Edoce and Rahs, that two twin sisters from last year's contest. They are playing a game together. Edoce draws a grid of arbitrary sizes and then Rahs computes the number of acute triangles that could be formed using points of the grid (In an acute triangle every angle is lower than 90 degree). For example, a 2×2 grid contains 8 acute triangles. Although they suck at card games, they are pretty good at geometry games. So they call the game ICPC (I Cant Play Cards). Can you compete with them? Write a program to compute the number of such triangles, given the grid's size.

Input

The input contains several test cases.
In the first line of input comes T $(0 < T < 32)$, the number of test cases.
Each test case is one line with two integer N and M $(1 \leq N, M \leq 100)$ indicating the number of points in each side of the grid.

Output

For each test print just one integer in one line, indicating the number of acute triangles.

Sample Input

```
2
1 1
2 2
```

Sample Output

```
0
8
```

C – Coordinate System

Description

Inefficient Coordinates Program, Cartesian section (ICPC) has created a problem in order to show how efficient this coordinates system is:
A 2D vector in Cartesian system is a pair of (a, b) where a and b are the displacements in x and y axes respectively. Adding vectors in this system is easy, you just need to sum up the values in each direction. For example, $(a, b)+(c, d)$ is equal to $(a+c, b+d)$. The result of the sum of vectors is also a vector. In some cases the sum of two vectors has a larger length than any of them and sometimes it gets smaller. Even there are cases that the sum of multiple vectors has length zero! Another interesting fact about adding vectors is that the order doesn't matter! You can sum them up in any order you want and the result would remain the same. Given a set of N 2D vectors in Cartesian system, find the number of subsets so that the sum of the vectors in that subset has length zero.

Input

The input contains several test cases.
In the first line of input comes T $(0 < T \leq 20)$, the number of test cases.
The first line of each test case contains $N \leq 40$, the number of vectors. The next N lines describe each vector. Each line contains two integers x and y where $|x|, |y| \leq 10$ and no vector has a zero length.

Output

For each test case, output on a single line containing the number of subsets with sum of zero length.

Sample Input

```
1
5
1 2
1 1
-1 -2
-2 -3
-1 -1
```

Sample Output

```
4
```

D – The Problematic Processor

Description

Laurel and Hardy have designed the new Intel Computer Processor for Charity (ICPC). This processor has 8 16-bit registers. Processor commands have 16 bits each.
Authorized commands for the processor are as followed. (Register 0 constantly contains a 0 and no value can be saved in it.)

Instruction	Bits				
	[15:12]	[11:09]	[08:06]	[05:03]	[02:00]
ADD	0000	rd	rs1	rs2	000
SUB	0001	rd	rs1	rs2	000
AND	0010	rd	rs1	rs2	000
OR	0011	rd	rs1	rs2	000
XOR	0100	rd	rs1	rs2	000
SL	0101	rd	rs1	rs2	000
SR	0110	rd	rs1	rs2	000
SRU	0111	rd	rs1	rs2	000

Instruction	Bits			
	[15:12]	[11:09]	[08:06]	[05:00]
ADDI	1000	rd	rs1	immediate
LD	1001	rd	offset	
ST	1010	rs	offset	
BZ	1011	rs	offset	
HALT	1100	—		

The operations executed in each command are explained below:

ADD rd rs1 rs2 − > rd = rs1+rs2
SUB rd rs1 rs2 − > rd = rs1 - rs2
AND rd rs1 rs2 − > rd = rs1 & rs2
OR rd rs1 rs2 − > rd = rs1 | rs2
XOR rd rs1 rs2 − > rd = rs1ˆrs2
SL rd rs1 rs2 − > rd = rs1<<rs2
SR rd rs1 rs2 − > rd = rs1>> rs2 (two's complement signed shift)
SRU rd rs1 rs2 − > rd = rs1>>rs2 (unsigned logical shift)
ADDI rd rs1 immediate − > rd = rs1 + Sign-extend (immediate)
LD rs offset − > rs = from [PC+Sign-extend (offset)] to [PC+Sign-extend(offset+1)]
ST rd offset − > from [PC+Sign-extend (offset)] to [PC+Sign-extend (offset+1)] = rd
BZ rs offset − > PC = (rs==0) ? PC+unsigned(offset) : PC + 1
HALT − > end of program

Note that the greater values are saved in the lesser bytes of the memory; e.g. for the command (LD 2 127), the 127th byte of the memory is saved in the 8 greater bits of register 2 and the 128th byte is saved in its 8 lesser bits. All registers have the initial value of 0. All the commands (with the exception of SRU) are executed with signed values and overflow is possible. No extra memory is allocated for the operations of "load" and "store" commands, these operations are instead executed in the commands' memory.
However, the processor has faced some problems: For each register i there is a chance of p_i that the data is not saved in the memory. Assuming you are given the values of p_i and the memories, determine the correct value (i.e. assuming all the previous operations to be done correctly) and expected value of each of the registers at the end of the execution.
Also note that "branch" and "store" commands are executed on the correct values of the registers and not the incorrect ones.

INPUT

The input contains several test cases.
In the first line of input comes T ($0 < T \leq 32$), the number of test cases.
For each test, the first line contains 7 numbers in the range of [0, 1] that represent the chance of registers 1 to 7 to be damaged. (Register 0 constantly contains a 0.)
The second line contains a number l (at most 1024) specifying the size of the memory in bytes. The input is followed in the next line by a string of length $2 * l$ (consisting of the numbers 0 to 9 and characters A to F or a to f), each representing 4 bits of the memory (in hexadecimal).

OUTPUT

For each test, if there existed any errors in the code, print the error expressions; otherwise print in 7 separate lines the expected values of registers 1 to 7 respectively, each with 5 places of precision. Also print the correct values of these registers in the following 7 lines.
Print "invalid operation" in case of receiving bad inputs for the commands. If access to invalid parts of the memory is requested, print "segment fault".

Sample Input

```
1
0.5 0.5 0.5 0.5 0.5 0.5 0.5
42
820184010650028004C00650028004C00650028004C00650028004C00650028004C00650028004C0C000
```

Sample Output

```
0.99512
1.43115
1.68848
0.00000
0.00000
0.00000
0.00000
13
21
21
0
0
0
0
```

E – **Weird Deal**

Description

The “ICPC Careless Packaging Company” (Confused?! it’s recursive!) has been selected to do a weird kind of delivery. A factory asked ICPC to deliver some packages from their storage to the factory. Each package has its own weight and there is a weight limit L for the truck as well. So the total sum of packages’ weight in the truck could not be more than L. On the other hand, ICPC’s trucks have only two compartments each. So each truck could not carry more than two packages at the same time.
The packaging company would charge the factory the multiplication of weights of the packages in each truck. So for delivering two packages with wight of 3 tons and 8 tons in one truck (assuming $8+3 = 12 \leq L$), the company charges the factory for $3 \times 8 = 24$ IRD (Iranian Dollars). The factory would not pay for the deliveries with just one package in the truck as the weight of the other packages would be zero.
According to the government’s laws, ICPC have to send the trucks at the same time and only once for each project. So they load all the trucks they need at the storage and send them to the factory, and that’s the end of the project. They can not reuse a truck in the same project.
In total, ICPC has M trucks, all of them with the same weight limit L. There are N packages in the factory’s storage.
ICPC might not be able to deliver all of the packages, but wants to maximize their profit. Now the task is to compute the maximum profit for the ICPC given the total number of trucks, their weight limit and the weights of each package in the storage.

Input

The input contains several test cases.
In the first line of input comes T ($0 < T \leq 32$), the number of test cases.
For each test case the first line comes with three integer N, M, L ($0 \leq N \leq 10^5$, $0 \leq M \leq 10^5$, $0 \leq L \leq 2 \times 10^4$), indicating the number of packages in the storage, the number of ICPC’s trucks and the weight limit of the trucks respectively.
In the next line there would be N space separated integers p_i ($0 \leq p_i \leq 10^4$) indicating the wight of the each packages in the storage.

Output

For each test print the maximum profit of ICPC for efficient delivery of the package.

Sample Input

```
2
4 2 6
4 1 3 2
4 1 6
2 1 4 5
```

Sample Output

```
11
8
```

F – Incredible Coding Products Company

Description

Incredible Coding Products Company (ICPC) owns lots of coding and decoding devices. They use them to code information in a way that detecting (and sometimes correcting) errors is possible.

According to Wikipedia, The Free Encyclopedia:

> "In coding theory, the BCH codes form a class of cyclic error-correcting codes that are constructed using finite fields. BCH codes were invented in 1959 by French mathematician Alexis Hocquenghem, and independently in 1960 by Raj Bose and D. K. Ray-Chaudhuri. The acronym BCH comprises the initials of these inventors' names. One of the key features of BCH codes is that during code design, there is a precise control over the number of symbol errors correctable by the code. In particular, it is possible to design binary BCH codes that can correct multiple bit errors. Another advantage of BCH codes is the ease with which they can be decoded, namely, via an algebraic method known as syndrome decoding. This simplifies the design of the decoder for these codes, using small low-power electronic hardware. BCH codes are used in applications such as satellite communications, compact disc players, DVDs, disk drives, solid-state drives and two-dimensional bar codes."

But currently ICPC has a problem. Their Multipliers are down and they cannot multiply polynomials without them. That's why they asked you to write a program to do that.
Given the coefficients of two polynomials find the coefficients of their multiplication.

Input

The input contains several test cases.
In the first line of input comes T $(0 < T \leq 10)$, the number of test cases.
Each test case starts with a line containing two integers n_a and n_b $(n_a, n_b < 100000)$, the number of non-zero coefficients in the two polynomials respectively.
The next line contains n_a pairs of integers like (a, i) where a is the coefficient of x^i (and is are given in increasing order). $0 \leq a < 100$, $0 \leq i < 100000$.
The next line contains n_b pairs of integers like (b, i) where a is the coefficient of x^i (and is are given in increasing order). $0 \leq b < 100$, $0 \leq i < 100000$.

Output

For each test case print the coefficients of the resulting polynomial in increasing order of term-degree.

Sample Input

```
2
2 2
1 0 1 1
1 0 1 1
2 2
2 0 1 1
3 0 1 1
```

Sample Output

```
1 2 1
6 5 1
```

G – The City of Karpeydi

Description

Instant Commute Planning Corporation (ICPC) is the contractor of a project in the city of Karpeydi: The city of Karpeydi has a number of buildings, between some of which exist streets. Each one of these streets is specified by a number in the range of $[0, 1]$, which represents the chance of heavy traffic in it. The chance of traffic in each street is independent of all other streets. Building a is accessible from building b if there exists a path between a and b with its chance of free-flowing traffic greater than or equal to a constant p (in the range of (0, 1]). We need to choose the least number of buildings from the city of Karpeydi and place ambulances in them, so that we have access to all buildings in the city in order to be able to send an ambulance as fast as possible to any building in case of fire.

Input

The input contains several test cases.
In the first line of input comes T $(0 < T < 16)$, the number of test cases.
For each test case, the first two lines contain $n \leq 25$ and $0 < p \leq 1$ representing the number of buildings and the least chance of free-flowing traffic between two buildings respectively. In each of the next n lines, you are given n numbers in the range of [0, 1], the number in row i and column j specifying the chance of heavy traffic between buildings i and j (and vice versa). If this number equals one, it means that the street is always jammed - or in fact does not exist. Assume the precision of each chance to be at most 10^3, i.e. each number is represented by at most 3 decimal places.

Output

For each test print one number which specifies the least number of buildings that need to be chosen for placing the ambulances.

Sample Input

```
2
5 0.5
0.000 0.2 1.0 1.0 0.6
0.2 0.000 0.2 0.5 1.0
1.0 0.2 0.000 1.0 0.6
1.0 0.5 1.0 0.000 1.0
0.6 1.0 0.6 1.0 0.000
9 0.6
0.000 0.4 1.0 1.0 0.5 1.0 1.0 0.3 1.0
0.4 0.000 1.0 0.3 1.0 1.0 1.0 1.0 1.0
1.0 1.0 0.000 0.4 1.0 1.0 1.0 1.0 1.0
1.0 0.3 0.4 0.000 0.15 1.0 1.0 1.0 1.0
0.5 1.0 1.0 0.15 0.000 0.29 0.2 1.0 1.0
1.0 1.0 1.0 1.0 0.29 0.000 0.4 1.0 1.0
1.0 1.0 1.0 1.0 0.2 0.4 0.000 0.3 1.0
0.3 1.0 1.0 1.0 1.0 1.0 0.3 0.000 0.1
1.0 1.0 1.0 1.0 1.0 1.0 1.0 0.1 0.000
```

Sample Output

```
2
2
```

Explanation for Sample Output

The buildings chosen for the first test case are buildings 2 and 5. For the second test case they are buildings 1 and 4, or buildings 4 and 8.

H – **Hardy-Ramanujan Number**

Description

According to wolfram.com:

> The smallest nontrivial taxicab number, i.e., the smallest number re-presentable in two ways as a sum of two cubes. It is given by $1729 = 1^3 + 12^3 = 9^3 + 10^3$.
> The number derives its name from the following story G. H. Hardy told about Ramanujan. "Once, in the taxi from London, Hardy noticed its number, 1729. He must have thought about it a little because he entered the room where Ramanujan lay in bed and, with scarcely a hello, blurted out his disappointment with it. It was, he declared, 'rather a dull number,' adding that he hoped that wasn't a bad omen. 'No, Hardy,' said Ramanujan, 'it is a very interesting number. It is the smallest number expressible as the sum of two [positive] cubes in two different ways' "
> ...

Now in the ICPC (International Conference of Positive Cubes), there is a special task which nobody can do. They have a weird algorithm to generate tags and seat numbers. In order to get the algorithm to work, they need to know how many pairs of positive cube exist which their summation would be the given number N.
Can you help the ICPC team to find it out?

Input

The input contains several test cases.
In the first line of the input comes T $(0 < T < 20000)$, the number of test cases.
Each test case is one line containing only one integer N $(1 \leq N \leq 2^{64})$.

Output

For each test print one line containing all the pairs (p_i, q_i) in this format:

$$N\texttt{=(}p_1\texttt{\^{}3+}q_1\texttt{\^{}3)=(}p_2\texttt{\^{}3+}q_2\texttt{\^{}3)=...}$$

You can also check the sample output to see the formatting. No space characters exist in the formatting.
All the pairs should be printed in the order that satisfy $p_i \leq q_i$ and $p_1 < p_2 < p_3 < ... < p_K$.
It is guaranteed that N could be written as the sum of two positive cubes in at least one way.

Sample Input

```
4
9
1729
4104
2746367559000
```

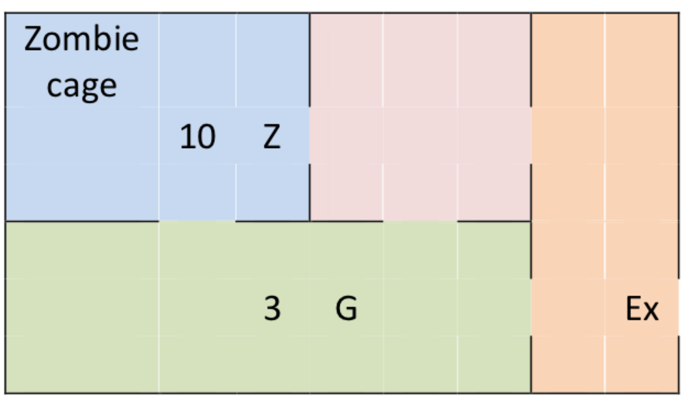
Sample Output

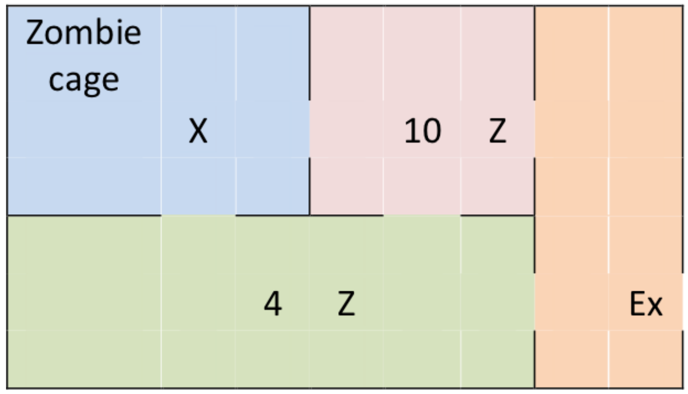
```
9=(1^3+2^3)
1729=(1^3+12^3)=(9^3+10^3)
4104=(2^3+16^3)=(9^3+15^3)
2746367559000=(246^3+14004^3)=(9755^3+12205^3)=(11110^3+11120^3)
```

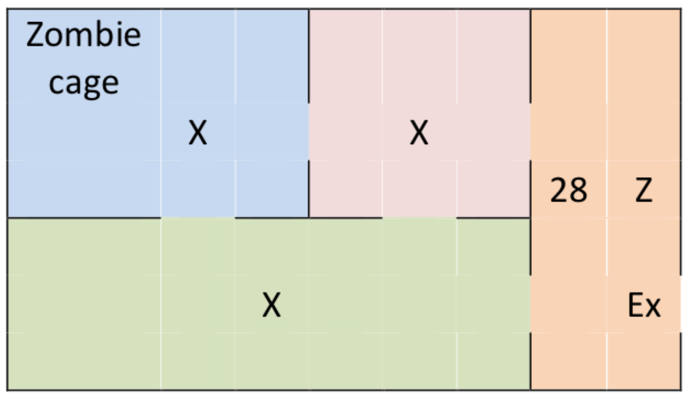
I – Zombies

Description

The "Institute of Changing Prot Chromosomes" (ICPC) has hired new zombie-terminator crew due to some dangerous experiments. There exists a room in this institute for keeping the zombies, and all the people in this room are either zombies or guards. If the zombies escape, the guards' duty is to make sure they are stopped before reaching the entrance to the processor center. The institute has a large number of rooms, some of which are connected to each other, and in each room - with the exception of the zombies' room - there exist an unlimited number of researchers. Due to security reasons, there is only one entrance to the processor center, the most important place in the institute.
Whenever the zombies enter a room, first every guard attacks a zombie, and as the result of this attack, both the zombie and the guard would be killed. Afterwards, if any living zombies still exist in the room, they each attack a researcher, transforming the researcher into a zombie as a result. Meanwhile, all other researchers in the room commit suicide in order not to become zombies. Hence, eventually the number of the zombies in the room would be doubled and all other people in the room would die. After all the living people in the room are transformed into zombies, the zombies go to the adjacent rooms in which there still exist living people. Assuming there are k rooms with living people adjacent to the current room, the zombies would be divided into k groups of equal size and move to these k rooms. If the number of the zombies is not divisible by k, the remainder of the zombies stay in the current room.
All the zombies move simultaneously and it is possible for two different groups of zombies to enter a new room at the same time. Also when the zombies reach the room which has the entrance to the processor center, they all exit to the processor center and not any other room - if they survive the attacks in the current room. Each room has a capacity of at most 10^9 zombies and all zombies trying to enter a filled-up room would be killed. An example of the procedure of the zombies' invasion is shown below. (Z: Zombie, G: Guard, X: Empty, Ex: Exit)







You are given the map of the institute (including the location of the processor center entrance and the labs) and the number of the guards in each room. You must find the most number of zombies that could be controlled by the guards in the institute (If there are more zombies than this number, they can exit to the processor center, and less zombies would be killed before reaching the exit.)

Input

The input contains several test cases.
In the first line of input comes T $(0 < T \leq 32)$, the number of test cases.
In each test, the first line contains $1 < N < 1000$, $0 < L < N$, and $0 < X < N$ representing the number of labs, the number of the zombies room, and the number of the lab that has the entrance to the processor center respectively. In the next N lines the information about each lab is entered as followed:
At the beginning of line $i+2$, you are given the number of guards in room i $(0 \leq Guards \leq 10^9)$. Next, you are given a number m as the number of the rooms adjacent to room i, followed by m numbers specifying these rooms.

Output

For each test case, print a line containing an integer representing the maximum number of zombies that could be stopped by the guards.

Sample Input

```
2
4 0 3
0 2 1 2
0 3 0 2 3
3 3 0 1 3
0 2 1 2
4 0 3
1 2 1 2
5 2 0 3
4 2 0 3
1 2 1 2
```

Sample Output

```
1
10
```

J – International Cord Pulling Contest

Description

The International Cord Pulling Contest (ICPC) is very special series of contests hosted by ShareCord every year. This year ShareCord's founders are pretty excited about the contest, because there is going to be an elders team formed from the old members of the federation. As they are pretty old and cranky, if they lose the game, they'll be pissed off and will leave the federation forever which is very bad for the federation's reputation. And it is also bad if they win the game since it discourages the younger contestants. The only way federation could get away is to arrange a perfect team with the exact strength as elders' team so they'll tie.
Arranging a perfect team is not that easy as there is another condition: In order to make it not suspicious, ShareCord need to make sure each member of the perfect team's strength is not less than a special amount called *suspicious level*.
There is no limit on the number of people in the team, but it should have at least one member. The strength of the team is the sum of strength of each team member and the strength of each member is an integer greater than zero. As each member will have a position number indicating which point of the cord he should hold at, the order is important too.
Assume the total strength of the elder's team is 3. So we have these three options to form the team:

- {3}
- {2, 1}
- {1, 2}
- {1, 1, 1}

Here for example, for suspicious level equal to 2, only those formations that all members' strengths are greater than or equal 2 would be accepted. So there is only one option:

- {3}

If the suspicions level is 1, all of those 4 combination would be accepted.

Now it is your task to write a program to compute the number of possible ways to form the perfect team given the strength of elder's team and the suspicious level.
You can assume the ShareCord community is huge and you can find infinite number of people with desired strength.

Input

The input contains several test cases.
In the first line of input comes T $(0 < T < 32)$, the number of test cases.
Each test case is one line with two integer N $(0 < N \leq 10^9)$ and K $(0 < K < 31)$, indicating the strength of the elder's team and the suspicious level respectively.

Output

For each test print the total number of ways to form the perfect team modulo $10^9 + 7$.

Sample Input

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

Sample Output

```
2
4
1
```

K – **Insane Cube Persecution Corporation**

Description

You wake up in a corner of a room without remembering how you got there. You see 6 other people and some doors. The strange thing is that there are doors on the walls (not so strange), the ceiling, and the floor! You remember a movie you saw a while back. The name had something like "Cube" in it?
Oh $#!%! This was real?!?! You see the logo of the company (ICPC) on a guys shirt! He should know everything about it. He starts explaining:
"The company is called Insane Cube Persecution Corporation (ICPC). We need to get from this room (room 1) to the last room (room n) but the number written on this key should be k when we want to exit room n". "What do you mean?" you ask. "The number on the key changes when we go to other rooms. It becomes the lowest common multiplier of the value before you enter and the number written on the wall of the room we enter, which might not be the same as the room number" he explains. You check the number on the key and it's equal to the number on the wall (remember that you are in room 1). "But what if the value remains the same when entering a room?" another guy asks. "Well,..." he hesitates, "we'll all die!".
Being a problem solver, you want to know how many ways are there to safely get out.

Input

The input contains several test cases.
In the first line of input comes T ($0 < T \leq 10$), the number of test cases.
Each test case begins with three numbers $2 \leq n \leq 2000$, $2 \leq m \leq 20000$, and $2 \leq k \leq 10^6$, separated with space. The next line contains n integers, $p_1, ..., p_n$ $1 \leq pi \leq 10^6$ the numbers written on the wall of each room. Then comes m lines describing the connection between rooms. Each line contains a sentence like: "There is a door to room v from room u.", where u and v are integers ($1 \leq u, v \leq n$, $u \neq v$). This means that you can go from room u to room v but you might not be able to come back (Yuhahhahaha!).

Output

For each test case, you should print a line containing a single integer, the number of paths to safely get out modulo 1000000007.

Sample Input

```
1
5 6 84
1 5 4 12 21
There is a door to room 1 from room 2.
There is a door to room 2 from room 5.
There is a door to room 1 from room 3.
There is a door to room 3 from room 5.
There is a door to room 1 from room 4.
There is a door to room 4 from room 5.
```

Sample Output

```
2
```