Problem A. Analyzing Login/Logout Records

Time limit 2000 ms
Mem limit 65536 kB

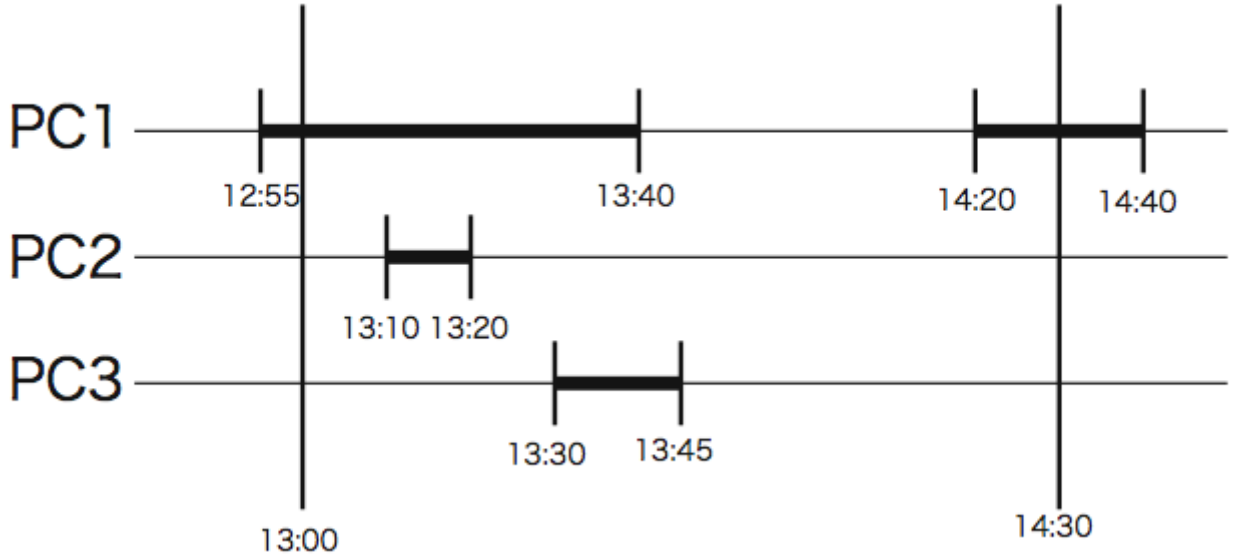
You have a computer literacy course in your university. In the computer system, the login/logout records of all PCs in a day are stored in a file. Although students may use two or more PCs at a time, no one can log in to a PC which has been logged in by someone who has not logged out of that PC yet.

You are asked to write a program that calculates the total time of a student that he/she used at least one PC in a given time period (probably in a laboratory class) based on the records in the file.

The following are example login/logout records.

- The student 1 logged in to the PC 1 at 12:55
- The student 2 logged in to the PC 4 at 13:00
- The student 1 logged in to the PC 2 at 13:10
- The student 1 logged out of the PC 2 at 13:20
- The student 1 logged in to the PC 3 at 13:30
- The student 1 logged out of the PC 1 at 13:40
- The student 1 logged out of the PC 3 at 13:45
- The student 1 logged in to the PC 1 at 14:20
- The student 2 logged out of the PC 4 at 14:30
- The student 1 logged out of the PC 1 at 14:40

For a query such as "Give usage of the student 1 between 13:00 and 14:30", your program should answer "55 minutes", that is, the sum of 45 minutes from 13:00 to 13:45 and 10 minutes from 14:20 to 14:30, as depicted in the following figure.



Input

The input is a sequence of a number of datasets. The end of the input is indicated by a line containing two zeros separated by a space. The number of datasets never exceeds 10.

Each dataset is formatted as follows.

> N M
> r
> record$_1$
> ...
> record$_r$
> q
> query$_1$
> ...
> query$_q$

The numbers N and M in the first line are the numbers of PCs and the students, respectively. r is the number of records. q is the number of queries. These four are integers satisfying the following.

> $1 \leq N \leq 1000, 1 \leq M \leq 10000, 2 \leq r \leq 1000, 1 \leq q \leq 50$

Each record consists of four integers, delimited by a space, as follows.

$t\ n\ m\ s$

s is 0 or 1. If s is 1, this line means that the student m logged in to the PC n at time t. If s is 0, it means that the student m logged out of the PC n at time t. The time is expressed as elapsed minutes from 0:00 of the day. t, n and m satisfy the following.

$540 \leq t \leq 1260, 1 \leq n \leq N, 1 \leq m \leq M$

You may assume the following about the records.

- Records are stored in ascending order of time *t.*
- No two records for the same PC has the same time *t.*
- No PCs are being logged in before the time of the first record nor after that of the last record in the file.
- Login and logout records for one PC appear alternatingly, and each of the login-logout record pairs is for the same student.

Each query consists of three integers delimited by a space, as follows.

$t_s\ t_e\ m$

It represents "Usage of the student m between t_s and t_e". t_s, t_e and m satisfy the following.

$540 \leq t_s < t_e \leq 1260, 1 \leq m \leq M$

Output

For each query, print a line having a decimal integer indicating the time of usage in minutes. Output lines should not have any character other than this number.

Sample

Input	Output
4 2 10 775 1 1 1 780 4 2 1 790 2 1 1 800 2 1 0 810 3 1 1 820 1 1 0 825 3 1 0 860 1 1 1 870 4 2 0 880 1 1 0 1 780 870 1 13 15 12 540 12 13 1 600 12 13 0 650 13 15 1 660 12 15 1 665 11 13 1 670 13 15 0 675 11 13 0 680 12 15 0 1000 11 14 1 1060 12 14 1 1060 11 14 0 1080 12 14 0 3 540 700 13 600 1000 15 1000 1200 11 1 1 2 600 1 1 1 700 1 1 0 5 540 600 1 550 650 1 610 620 1 650 750 1 700 800 1 0 0	55 70 30 0 0 50 10 50 0

Problem B. Finding Nemo

Time limit 1000 ms
Mem limit 65536 kB

Boudreaux and Thibodeaux are just returning from watching Finding Nemo and are finding themselves pretty hungry after watching all those fish swim around for a couple of hours. Like the true Cajuns that they are, they jump into their pickup and head on over to the local bayou. Upon arriving Boudreaux realizes that in their mad hunger rush, they have completely forgotten their fishing poles and tackle. Boudreaux yells out to Thibodeaux, "Hey couyon, you forgot about dem poles!" and Thibodeaux replies "Don't worry, I got me some fish sticks out in the truck." Boudreaux later finds out that "fish sticks" are really dynamite, to which he replies "Mais fool, now how you suppose we gonna know where to place dem sticks to catch some fish?" at which point Thibodeaux then breaks out his fish finder and laptop and whips up a program that will tell them just that.

Given the position of all the fish in the bayou and the spot Boudreaux and Thibodeaux want to drop their dynamite (after lighting the fuses of course), you are to write a program that will tell how many fish they will kill, ahem, I mean catch. Each dynamite stick has a certain fuse length that determines at which depth it will blow up. Any fish within a one unit radius of the dynamite when it blows up is as good as fish fried. Keep in mind that the fish never move from their location, and the dynamite sticks fall straight to the bottom of the bayou.

Input
Input to this problem will consist of a (non-empty) series of up to 100 data sets. Each data set will be formatted according to the following description, and there will be no blank lines separating data sets.

A single data set has 4 components:

1. Start Line - A single line:
   START L W D

   where (1 <= (L,W,D) <= 20). L is the length of the bayou along the x-axis, W is the width of the bayou along the y-axis, and D is the depth of the bayou on the z-axis.
2. Dynamite List - A single line containing a space-separated list of 1 to 10 data elements. Each element contains the location and fuse length of a single stick of dynamite as it is dropped into the bayou formatted as:
   x,y,f

x and y give the surface coordinates of the drop where (0 <= x <= L) and (0 <= y <= W). f is the length of the fuse and is in the range (0 <= f <= 30).

3. Fish List - A single line containing a space-separated list of 1 to 15 data elements. Each element represents the location of a fish in the bayou formatted as:
   x,y,z

   where (0 <= x <= L), (0 <= y <= W), and (0 <= z <= D), where z = 0 indicates the fish is at the surface of the water.
4. End line A single line:
   END

After the last data set, there will be a single line:
ENDOFINPUT

Note:
All numeric values will be given as integers.
The dynamite drops at a constant speed from the top of the bayou (z = 0).
The fuse burns up one unit of its length in precisely the amount of time it takes the dynamite to sink one unit deeper into the bayou.
If the dynamite reaches the bottom of the bayou before the fuse runs out, it will stay there until it detonates.
Multiple fish will not occupy the same position.
Fish are killed if their distance from any dynamite explosion is <= 1.

Output

For each data set, there will be exactly one line of output. The output will be a phrase stating how much fish Boudreaux and Thibodeaux will be frying up tonight.

If they blow up at least one fish, the following phrase will be printed:

AIEE, I got N fish, me!

where N is the number of fish blown up. If they don't blow up any fish, the following phrase will be printed:
None of dem fish blowed up!

Sample

Input	Output
START 5 5 5 1,1,1 2,2,2 3,3,3 4,3,0 4,4,4 3,0,2 2,1,3 3,3,3 END START 2 3 4 1,1,10 1,1,1 0,2,2 0,0,1 END ENDOFINPUT	AIEE, I got 1 fish, me! None of dem fish blowed up!

Problem C. Table Legs

Time limit 1000 ms
Mem limit 65536 kB

A table with four legs may rock, even on a flat surface, if its legs are not all the same length. Interestingly, regardless of how many legs have differing lengths, it is always possible to saw an amount from some legs so as to make the table sit level on a flat surface without rocking.
Your job is to generalize this approach to a table with many legs equally spaced around the perimeter of a round table. You are to determine the total length of legs to cut so as to have the table sit level without rocking on a flat surface with not necessarily every leg touching the ground.

Input
Input consists of data for a number of tables. For each table, a line will give an integer t, between 3 and 50, indicating the number of legs on the table. t subsequent lines will give, in order around the table's circumference, the lengths of the legs in millimetres. Each leg is perpendicular to the table top. A line containing 0 follows the data for the last table.

Output
Pick a strategy that cuts the least total length from all legs and print this amount as an integer number. Print a blank line between tables.

Sample

Input	Output
3 2000 3000 4000 4 2000 2000 1999 2001 5 2000 2000 1999 2001 1999 0	3000 4 1

Problem D. Kingdom

Time limit 2000 ms
Mem limit 65536 kB

King Kong is the feared but fair ruler of Transylvania. The kingdom consists of two cities and N < 150 towns, with nonintersecting roads between some of them. The roads are bidirectional, and it takes the same amount of time to travel them in both directions. Kong has G < 353535 soldiers. Due to increased smuggling of goat cheese between the two cities, Kong has to place his soldiers on some of the roads in such a way that it is impossible to go from one city to the other without passing a soldier. The soldiers must not be placed inside a town, but may be placed on a road, as close as Kong wishes, to any town. Any number of soldiers may be placed on the same road. However, should any of the two cities be attacked by a foreign army, the king must be able to move all his soldiers fast to the attacked city. Help him place the soldiers in such a way that this mobilizing time is minimized.
Note that the soldiers cannot be placed in any of the cities or towns. The cities have ZIP-codes 95050 and 104729, whereas the towns have ZIPcodes from 0 to N - 1. There will be at most one road between any given pair of towns or cities.

Input
The input contains several test cases. The first line of each test case is N, G and E, where N and G are as defined above and E < 5000 is the number of roads. Then follow E lines, each of which contains three integers: A and B, the ZIP codes of the endpoints, and φ, the time required to travel the road,φ < 1000. The last line of the input is a line containing a single 0.

Output
For each test case in the input, print the best mobilizing time possible, with one decimal. If the given number of soldiers is not enough to stop the goat cheese, print "Impossible" instead.

Sample

Input	Output
4 2 6 95050 0 1 0 1 2 1 104729 1 95050 2 1 2 3 3 3 104729 1 4 1 6 95050 0 1 0 1 2 1 104729 1 95050 2 1 2 3 3 3 104729 1 4 2 7 95050 0 1 0 1 2 1 104729 1 95050 2 1 2 3 3 3 104729 1 2 1 5 0	2.5 Impossible 3.0

Problem E. FlashGet

Time limit 1000 ms
Mem limit 65536 kB

You are downloading some things using a software (Flashget, maybe), but suddenly you have to go away for something. So, you need a program to calculate when the tasks will complete.

Now you know every download's speed, size and max speed. When a task is over, the bandwidth used is distributed by other tasks. The speed of one task can never go beyond the max speed of this task, and all tasks' speed can never be larger than the total bandwidth.

Input

There are multiple cases in the input.

The first line of each case contains two integers, n and t (n<=100). n is the number of tasks, and t is the total bandwidth. There follows n lines, one line has three integers, means the size of the download file, the initial speed and the max speed. The input promises the sum of tasks' speed equals to the total bandwidth, the speed is not higher than the max speed.

The input is terminated by a zero.

Output

For each case first print "Case %:" in one line, % is the number of cases.
Then print n lines for n tasks, like this:
NO*:#s
* means the number of task, for the sequence of input; # means the finish time of the task.

Sample

Input	Output
3 65 100 20 30 200 30 30 300 15 30 0	Case 1: NO1:5.000s NO2:6.667s NO3:12.500s

Hint

The bandwidth will never change if no task is finished.

When a task finished, the bandwidth is distributed by this rule:
Every unfinished task which does not reach the max speed gets the same bandwidth; the total

bandwidth can not be overflowed; every task's speed can not overflow its max speed; if there is bandwidth can be used, distribute it.

Problem F. “Roman” corridor

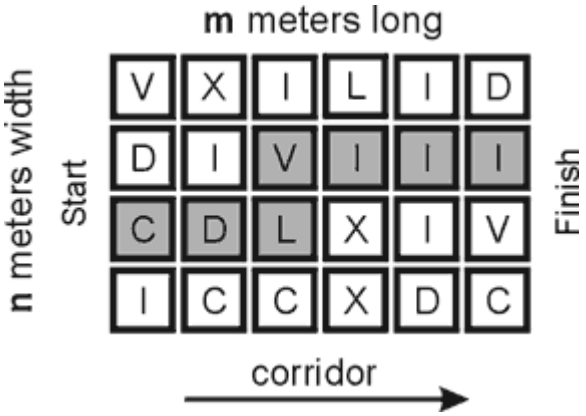
Time limit 1000 ms
Mem limit 65536 kB

Let’s remind the notation of Roman numerals. The notation is for natural numbers from 1 to 3999. Capital Latin letters ‘I’, ‘V’, ‘X’, ‘L’, ‘C’, ‘D’, ‘M’ and their combinations are used to represent so called *atomic* numbers (see the table below).

1 – I	40 – XL	500 – D
4 – IV	50 – L	900 – CM
5 – V	90 – XC	1000 – M
9 – IX	100 – C	
10 – X	400 – CD	

To put down a number **N** it is necessary to find the greatest *atomic* number **K** which is not greater then **N**. The Roman notation of the found number **K** is put down, and the process is repeated for **(N-K)**.

The Roman numerals are put down from left to right without spaces. Thus, the number **999** in the Roman notation is **CMXCIX** (but not **IM**, as somebody may think). You need to pass through a rectangular corridor. The corridor is **n** meters width and **m** meters long ($1 \le n, m \le 15$, $n*m \le 100$). It is laid out by square tiles. Each tile is 1 meter width and has a ‘Roman’ symbol on it: ‘I’, ‘V’, ‘X’, ‘L’, ‘C’, ‘D’ or ‘M’. When passing the corridor, you move from one tile to another. From the current tile you may only move to one of adjacent tiles, vertically or horizontally (but not across). You start at the left and end at the right (see the picture below).



Can you pass through the corridor so that the sequence of symbols on the tiles composing your path was a correct number in the Roman notation? Among all possible solutions you need to find

the minimal number.

Input

The first line contains numbers n and m, separated by one or more spaces. Each of the next n lines consists of m characters describing tiles.

Output

The output contains one line with the found Roman number or the word **NO** if it is impossible to pass through the corridor in the required way.

Sample

Input	Output
4 6 VXILID DIVIII CDLXIV ICCXDC	CDLVIII

Problem G. Movie Theatre Madness

Time limit	1000 ms
Mem limit	1572864 kB
Code length Limit	50000 B
OS	Linux

A group of friends have gone to watch the first day first show of an awesome new movie. However, since they did not book the tickets well in advance, they have ended up with crazy seats. To be more specific, rather than getting seats such that all the friends are seated in the same row, they have ended up with seats such that all of them are seated in the same column! Now this is very inconvenient since they won't be able to chat with each other during the movie or have any kind of fun, but they are okay with this since they'd rather watch the movie like this, than not watch the movie at all.

But there is another problem apart from this. Now we know that all the friends are seated in a single column, one behind the other. Since all of them reached the theatre just in time for the movie, they rushed and occupied the first of the booked seats that they could find. However all the friends have different heights, and due to the lack of planning, there is no guarantee that a shorter person is always seated in front of a taller person. But this would mean that the shorter person would struggle to see the screen throughout the movie!

But the movie has started and it's too late now to do anything.

What you need to do is the following: For every person, find the height of the closest person seated in front of him/her who is blocking his/her view (that is, the person closest in front with a greater height). If no such person exists, take this height as 1. Print the product of all such values modulo 1000000007.

Input

On the first line you have a single integer N ($2 <= N <= 10^5$), the total number of friends.

This is followed by N space separated integers a_1, a_2, ... a_N, which correspond to the height of the people from back to front. That is, a_1 is the height of the person seated on the last row, a_2 is the height of the person seated on the second last row (just in front of a_1) and so on, up to a_N which is the height of the person seated right at the very front ($1 <= a_i <= 10^9$). Note that all these integers are distinct.

Output

On a single line, output the result. (For every person, find the height of the closest blocking person. This value is 1 if no such person exists. Print the product of all these values modulo 1000000007).

NOTE: By closest blocking person to a_i we mean find a_j, such that $a_j > a_i$, $j > i$, and $(j-i)$ is minimum. (Please look at sample test cases for further clarity.)

Example

Input #1:

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

Output #1:

```
16
```

Input #2:

```
5
9 8 3 5 7
```

Output #2:

```
35
```

Input #3:

```
10
30 10 50 70 11 60 20 80 31 12
```

Output #3:

```
999962375
```

Explanation

Input #1:

- blockingHeight(5) = 1 (since 5 is the tallest, no one is blocking him)
- blockingHeight(2) = 4
- blockingHeight(1) = 4 (although 3 is also taller that 1, 4 is closer to 1)
- blockingHeight(4) = 1 (No one blocking 4)
- blockingHeight(3) = 1 (No one blocking 3)
- Answer=1*4*4*1*1=16

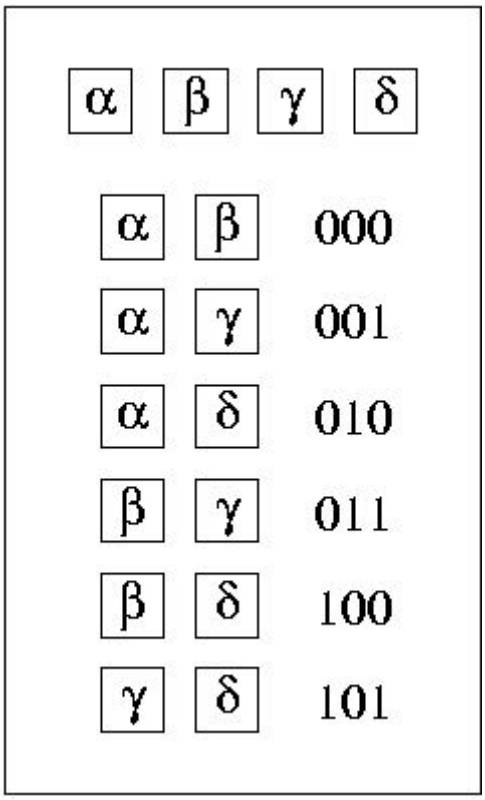
Problem H. California Jones and the Gate to Freedom

Time limit 1000 ms
Mem limit 30000 kB

California Jones (the sister of famous Indiana Jones) once again faced a seemingly intractable problem. Her only hope was in you. She knew you were a computer scientist and you might have a clue.

Jones calls you on the video-phone and tells you the facts: she walked into a trap and now stands in front of a huge gate. On the left side strange signs can be seen while n stones lie on the right side. In front of the gate there are exactly n/2 holes. Says Jones, "I suppose I have to take exactly half of the stones from the right side and put them into the holes." Ancient writings confirm her conjecture. According to the writings it does not matter which hole a stone is placed into. It is only important that the right stones are chosen.

Nearby, Jones found a stone board, too, but was unable to interpret. It made sense to you though. It was a hint on how to sort the various possibilities of chosing n/2 stones.
But you couldn't yet figure out about the zeros and the ones. So you asked Jones who replied that "the same symbols I saw on the left side of the gate - only they were somewhat longer sequences. But I haven't met such a primitive civilization yet."



Now everything was clear to you: the symbols were the representation of a binary number - and it indicated which stones to choose. Simply ingenious! Jones was enthusiastic about you.

But it was impossible for Jones to calculate for a given binary number the corresponding stones. So she instructed you to write a program to solve the task and help her through the gate to freedom. Five hours later, she would call back.

Take a thorough look at the figure to the left depicting the stone board, as well as the sample input and output, to figure out how to solve Jones' problem.

Input

The input contains several testcases. Each starts with the number of stones n. Input is terminated by n=0. Otherwise, n is even and 2<=n<=32. The next n integers identify the stones. A test case is further subdivided into k (sub-) test cases, k being the next number in the input file. Then follow k times a bit string b (encoding a non-negative integer) and n/2 distinct integers identifying the set of chosen stones. No invalid stones will be chosen and the length of b will not exceed 30.

Output

For each (sub-) test case generate a line containing TRUE, if the chosen stones may be laid into the holes, and FALSE otherwise.

Sample

Input	Output
4 12 50 74 34 1 00 50 12 8 45 23 86 43 90 76 12 74 2 111001 86 43 90 74 010001 45 86 43 90 4 12 50 74 34 2 101 34 74 110 34 74 0	TRUE TRUE FALSE TRUE FALSE