A – Grades

Description

Professor Snape has just marked the final exam papers and wants to know if the distribution of grades is satisfactory or not. He first sorted all papers by their grades and then took the best and worst marks and took their average. Let's call his number A. Then he took the average of all the grades. Let this number be called B. If and only if the absolute difference between A and B is less than 1, he considers the distribution to be satisfactory.
Write a program to help professor Snape.

Input

The first line contains the number of test cases T.
Each test case consists of two lines, the first line is a single integer, $n \leq 50000$, the number of students and the second line has n space separated non-negative integers smaller than 101, the grades students received.

Output

For each test case print *"Yes"* (without quotations) if the distribution is satisfactory and *"No"* (without quotations) otherwise.

Sample Input

```
2
3
1 2 3
3
1 10 100
```

Sample Output

```
Yes
No
```

B – DotB

Description

The new DotA game is coming out soon. All the fans around the world are waiting to try the new version of DotA: "DotB".
There's a lot of pressure on the developers of the game to add new heroes to it. They come up with ideas but not all of them are worth implementing. For one particular hero they are working on, they asked you to write a program to analyze one of its abilities. The hero's name is "Cave Man", and his ability is to summon a crow to attack a line of enemies.
The line has n enemies with hitpoint $h_1h_2...h_n$. The crow always starts at the first enemy and attacks each one and goes to the next. It attacks exactly $n + 5$ times in total, and if it reaches the end of the line, it continues the attack from the other end of the line.
The crow has the damage C. Which means C would be deducted from the enemy's hitpoint if it receive an attack. If an enemy's hitpoint hits zero or less, well ... he's dead.
If the crow's attack causes death of an enemy, the crow reverses its movement direction.
Now it's your task to find which enemy receives the last attack of the crow, given the enemies' hitpoint and the crow's damage.

Input

The first line contains the number of test cases T.
Each test case begins with two space-separated integers n ($2 \leq n \leq 32$) and C ($32 \leq C \leq 128$). Then comes n integers indicating the initial hitpoint of the enemies. The ith integer indicates the hitpoint of ith enemy in the line.

Output

For each test case, print a single line containing the serial number of the enemy the crow finally attacks.

Sample Input

```
2
3 100
200 150 200
8 80
200 100 100 100 100 80 160 200
```

Sample Output

```
2
3
```

C – Sticks and Carrots(Chomagh va Havij)

Description

Rabbits have attacked Georgia and they are eating all the carrots. Farmers try to protect their crops and Hershel Greene, one of the farmers, has special carrots in his farm which are very expensive. So he puts up some security gates all around his farm. These gates are in the form of pillars which knock out any living creature that passes between two pillars by throwing a stick at them and makes it unconsiouse. (Farmers are protective of animals and will not kill them.)
Every noon at 12 o'clock the gates are disabled and Hershel can visit the farm and he is only able to exit the farm at exactly 12:30. Hershel is lazy and he wants to only move in a straight line from any carrot to another but he cannot exit the farm.
After some months, some of the carrots have been wasted due to insects' attacks. Hershel wants to sell some of the extra gates to lower the costs by making the farm smaller and he needs your help. The area of the farm has straight relation to the cost.
Note that he is lazy and he cannot leave the farm while visiting the carrots.
Even if there are no intact carrots left, Hershel will not sell all the gates because he still, has a farm to keep!

Input

The first line contains the number of test cases T.
First line of each test case contains two space separated integer numbers of security gates, G ($0 \leq G \leq 1000$) and number of carrots, C ($0 \leq C \leq 50$). Next G Lines contain security gate's positions (x and y which are integers) and they are ordered clockwise.
Afterward, the next C Lines contain carrots (also x and y which are integers). You can assume the absolute values of any number does not exceed 10000.

Output

Print in a single line, the area of smallest farm which contains all of the carrots (accurate to two digits to the right of the decimal point).

Sample Input

```
1
5 2
8 9
0 -7
-8 -7
-8 1
-8 9
-4 -3
-1 -5
```

Sample Output

```
128.00
```

D – Son of Durin

Description

Well, this story is about *'the company'*, Dwarves company. As you know, *Thorin*, king of Erebor, have become so greedy and wants to seek and collect golden coins as much as he can. But he knows that *winter is coming* (sound of drums...), and he doesn't have enough time for searching gold in peace of mind (maybe the Dragon will wake up or... nobody knows!).
Now Thorin have found a map that shows the paths inside the mountain, and also the caves which contain a lot of golden coins (fortunately, the map shows how many coins each cave has! what a mysterious map...).
The loyal blood which is pumping through Thorins' vein, gives him a magical ability to *Teleport*! Yes he can teleport by hands of gods and wings of angels. Oh wait! He just said to me he can just teleport from some specific places to some other places in the map! Okay Thorin, thanks for correcting me!
Thorin gave me a map which I will share it with you. And I will tell you how much time is left until winter. The map is a $N \times M$ grid, normal cells are shown with a *'.'* which Thorin can pass through, *'#'* indicates the cells which Thorin can't go there or cross them by walking, '^' denotes the teleport sites. He can go from each '^' to the other '^' in the map in a single unit of time. The numbers and uppercase english letters indicates the locations of corresponding cave in hexadecimal form. For example 0, 3, and B are indicating the first, fourth and twelfth cave respectively (numbers are zero-based). Thorin is located with a *'d'* on the map.
In each unit of time, Thorin can go from one cell to exactly one non-sharped adjacent cell in up, right, left, or down directions. When Thorin moves into a '^' cell, he *can* teleport to other '^'s on the map in a unit of time (note that he can also move to adjacent cells if possible). When he reaches a cave containing golden coins, he can instantly grab all of the coins in a moment (it takes no time).

Input

First line of input contains one integer $T(1 \leq T \leq 10)$, which is the number of test cases.
For each test case:

The first line: four space separated integers N, M $(1 \leq N, M \leq 500)$, *tp-limit* $(0 \leq tp - limit \leq 10^5)$ and *time-limit* $(0 \leq time - limit \leq 10^9)$.

The next N lines: Each contains M characters which i^{th} line shows the i^{th} row of the map.

The last line: contains space separated integers, the number of golden coins in each cave corresponding to the number of that cave in the map, *cave-values* $(0 \leq cave - values \leq 10^9)$.

tp-limit is the maximum number of times that Thorin can use his special teleport ability. For example if *tp-limit* is one, he can teleport at most one time to another teleport site. Please help Thorin find the maximum amount of coins he can collect within *time-limit* units of time and using at most *tp-limit* teleports. You can assume that there are less than 16 caves in a map.

Output

For each test print a single line containing the maximum amount of golden coins that Thorin can collect.

Sample Input

```
1
6 4 1 15
#d.^
##..
....
0...
####
1.^.
100 1000
```

Sample Output

```
1100
```

E – Halting Machine

Description

"In computability theory, the halting problem is the problem of determining, from a description of an arbitrary computer program and an input, whether the program will finish running or continue to run forever.
Alan Turing proved in 1936 that a general algorithm to solve the halting problem for all possible program-input pairs cannot exist. A key part of the proof was a mathematical definition of a computer and program, which became known as a Turing machine; the halting problem is undecidable over Turing machines. It is one of the first examples of a decision problem.
The halting problem is historically important because it was one of the first problems to be proved undecidable. (Turing's proof went to press in May 1936, whereas Alonzo Church's proof of the undecidability of a problem in the lambda calculus had already been published in April 1936 (Church, 1936).) Subsequently, many other undecidable problems have been described; the typical method of proving a problem to be undecidable is with the technique of reduction. To do this, it is sufficient to show that if a solution to the new problem were found, it could be used to decide an undecidable problem by transforming instances of the undecidable problem into instances of the new problem. Since we already know that no method can decide the old problem, no method can decide the new problem either. Often the new problem is reduced to solving the halting problem. (Note: the same technique is used to demonstrate that a problem is NP complete, only in this case, rather than demonstrating that there is no solution, it demonstrates there is no polynomial time solution, assuming $P \neq NP$).
For example, one such consequence of the halting problem's undecidability is that there cannot be a general algorithm that decides whether a given statement about natural numbers is true or not. The reason for this is that the proposition stating that a certain program will halt given a certain input can be converted into an equivalent statement about natural numbers. If we had an algorithm that could find the truth value of every statement about natural numbers, it could certainly find the truth value of this one; but that would determine whether the original program halts, which is impossible, since the halting problem is undecidable.
Rice's theorem generalizes the theorem that the halting problem is unsolvable. It states that for any non-trivial property, there is no general decision procedure that, for all programs, decides whether the partial function implemented by the input program has that property. (A partial function is a function which may not always produce a result, and so is used to model programs, which can either produce results or fail to halt.) For example, the property "halt for the input 0" is undecidable. Here, "non-trivial" means that the set of partial functions that satisfy the property is neither the empty set nor the set of all partial functions. For example, "halts or fails to halt on input 0" is clearly true of all partial functions, so it is a trivial property, and can be decided by an algorithm that simply reports "true." Also, note that this theorem holds only for properties of the partial function implemented by the program; Rice's Theorem does not apply to properties of the program itself. For example, "halt on input 0 within 100 steps" is not a property of the partial function that is implemented by the program—it is a property of the program implementing the partial function and is very much decidable.

> Gregory Chaitin has defined a halting probability, represented by the symbol Ω, a type of real number that informally is said to represent the probability that a randomly produced program halts. These numbers have the same Turing degree as the halting problem. It is a normal and transcendental number which can be defined but cannot be completely computed. This means one can prove that there is no algorithm which produces the digits of Ω, although its first few digits can be calculated in simple cases.
> While Turing's proof shows that there can be no general method or algorithm to determine whether algorithms halt, individual instances of that problem may very well be susceptible to attack. Given a specific algorithm, one can often show that it must halt for any input, and in fact computer scientists often do just that as part of a correctness proof. But each proof has to be developed specifically for the algorithm at hand; there is no mechanical, general way to determine whether algorithms on a Turing machine halt. However, there are some heuristics that can be used in an automated fashion to attempt to construct a proof, which succeed frequently on typical programs. This field of research is known as automated termination analysis.
> Since the negative answer to the halting problem shows that there are problems that cannot be solved by a Turing machine, the Church–Turing thesis limits what can be accomplished by any machine that implements effective methods. However, not all machines conceivable to human imagination are subject to the Church–Turing thesis (e.g. oracle machines). It is an open question whether there can be actual deterministic physical processes that, in the long run, elude simulation by a Turing machine, and in particular whether any such hypothetical process could usefully be harnessed in the form of a calculating machine (a hypercomputer) that could solve the halting problem for a Turing machine amongst other things. It is also an open question whether any such unknown physical processes are involved in the working of the human brain, and whether humans can solve the halting problem (Copeland 2004, p. 15)."
> –Wikipedia

However, you can solve the halting problem for a program in language X under certain assumptions.
Each program in language X consists of N goto statements with the following structure (there are no white-spaces in conditions and the spaces are only after ':', ',', and the goto keyword):
goto l1: c1, l2: c2, . . . , lk: ck;
This means that the program goes to line number $li \geq 0$ if condition ci is true, $0 < i \leq k$;
Assuming that the conditions are independent and possible with some input, your task is to find out if the program halts, and if so, in how many steps. A program halts if and only if it reaches line number N.

Input

The first line of the input contains one integer T($0 \leq T \leq 11$), the number of test cases.
Each test case is a program:

The first line : integer N($0 \leq N \leq 1000$) the number of lines in that program.
The next N lines: each contains exactly one goto statement.

Output

For each test case, output the number of steps in the worst case or *"infinity"* (without quotations) if the program does not halt in some case.

Sample Input

```
2
2
goto 1:  a>b;
goto 0:  10<x;
3
goto 2:  a<b, 1:  b<a;
goto 3:  a>0;
goto 1:  b>0;
```

Sample Output

```
infinity
3
```

F – "Oland must not fall"

Description

It's about a month since the sixth army of the evil Miniophers has invaded Olandica. Their troops are now spread all over Oland, an strategic city in the west of Olandica, and have made a lot of casualties to the civilians. Fortunately, a few hospitals are still held by the Olanders, but there is a shortage of Unophine, a medication needed to cure damages caused by a biological weapon the wicked Miniophers have been using. To remedy this, the Olanders has dug a number of tunnels connecting some of the hospitals. This way, a Unophine pack may be delivered from a hospital to another if there is a tunnel connecting them. Note that since the tunnels are not that wide, each time only a single Unophine pack may be transfered via a tunnel.

Three times a day, each hospital contacts the operations HQ and announces the number of Unophine packs needed, and the number of packs it currently stores. Then, HQ comes up with a plan (if possible) to distribute the packs to match the hospitals' needs. The cost of a plan is the number of packs to be transferred. As an example, consider three hospitals A, B, and C which store 3, 2, and 3 packs respectively. Suppose there are two tunnels between A and B, and between B and C. If the hospitals need 4, 3, and 1 packs, a distribution plan (of cost 3) would be to transfer one pack from B to A, and then two packs from C to B.

The problem is, given the tunnels and the current pack storage, determine the minimum cost of a distribution plan to match the needed packs.

Input

There are $T < 32$ test cases in the input. Each test case starts with a pair of two integers L and R ($1 \leq L, R \leq 128$) which are the number of hospitals and the number of tunnels respectively. The hospitals are numbered from 1 to L. Each of the next L lines contains two integers s_i and n_i ($1 \leq s_i, n_i \leq 10000$) which specify the number of packs the i^{th} hospital stores and the number of packs it needs. Each of the next R lines contains two integers i and j between 1 and L which indicates that there is a tunnel between the hospitals i and j.

Output

For each test case, output the minimum cost of the distribution plans on a single line. if it is not possible to find a plan, output `-1`.

Sample Input

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

Sample Output

```
2
-1
```

G – Circular Board Game

Description

In the city of Bigwood there is a popular board game named Expo in which the board has a circular shape with N cells (1 to N). This game, same as many other board games, is played using tokens and a die. Players take turn to roll a K-sided die but the movement based on the number rolled is not that simple. The rule is that, assuming the side S has appeared, you move your token expo(S) cells forward. Where expo(S) = $S^{(S-1)^{(S-2)^{\cdots^{(2)^{(1)}}}}}$. As the board is circular, the movement will continue when we reach to the last cell. The goal is to put your token in the last cell starting from the first one. (Ofcourse not in the middle of a movement)
Brad and Pete are playing this game. But as you may have noticed, for larger amounts of S, it might take a lifetime (or even much much more!) to move the token from one cell to the next one for expo(S) times. Actually, their grandfathers couldn't finish the game they had started when they were young in the traditional way. So Brad and Pete are playing on behalf of their grandfathers and expect you to help.
Find the destination cell for a given side S on the die, starting from Mth cell.

Input

The first line of input contains one integer T ($1 \leq T \leq 256$), which is the number of test cases.
For each test case: three space separated integers S ($1 \leq S \leq 10^9$), M, and N ($1 \leq M \leq N \leq 10^9$).

Output

For each test, print a single line containing the place token lies on the board after the movement.

Sample Input

```
3
2 1 39
94 250 265
5 123 123456789
```

Sample Output

```
3
24
16317757
```

H – Blocks&Balls

Description

After successful sending of a monkey into the space, our country has proudly sent a genius five-year-old boy, named Masoud! He has taken some of his toys with him. His favorite toy is Blocks&Balls, a cuboid container in which he places several small cuboid blocks and balls. Since there is no gravity out there, the blocks and balls do not fall into the bottom of the container. After placing a little blocks and balls, Masoud wondered, "if I pour some water in the container, how high the level of the water would be?" Obviously, he cannot experiment that in space, since the water does not fill the container from the bottom, but he can write a computer program to calculate it!

Input

There are $T < 64$ test cases in the input. The first line of each test case contains 3 floating point numbers w, ℓ $(0 < w, \ell < 100000)$ which are the width and the length of the container, and v $(0 < v < 10^{13})$, the volume of the poured water. You may assume the height of the container to be infinite. Two integers follow in the same line which are m $(1 \leq m \leq 100000)$, the number of the blocks, and n $(1 \leq n \leq 100000)$, the number of the balls. The next m lines describe the position and the size of the blocks using four floating point numbers z $(0 < z < 100000)$, the height of the center of the block, a $(0 < a < w)$, b $(0 < b < \ell)$, and c, the width, length, and height of the block. The next n lines describe the position and the size of the balls using two floating point numbers $z(0 < z < 100000)$, the height of the center of the ball, and $r(0 < 2r < \min(w, \ell))$, the radius of the block. You may assume that the input is correct, i.e., no two objects (blocks and balls) intersect and all objects fit inside the container.

Output

For each test case, output the height of the water level in a single line. An answer with absolute error less than 10^{-4} or relative error less than 10^{-6} will be accepted.

Sample Input

```
1
1 1 1 1 1
1.5 0.2 0.3 0.4
0.5 0.5
```

Sample Output

```
1.537869
```

I – Minions and the rooms

Description

N minions live in the basement of Gru's house and are numbered from 1 to N. There are M rooms with infinite capacity in the basement and each room has two states: available or unavailable. Gru will reverse the state of consecutive rooms every day. Minions can only live in available rooms. And each available room should contain at least one Minion.
Gru is interested in how many different ways to arrange creatures in rooms. At the beginning, all rooms are available. In the following D days, Gru will ask you the number of ways after reversing. Two ways A and B are regarded as the same if for any room in A, there exists a room in B that the creatures in these two rooms have the same set of numbers. In other words, rooms are indistinguishable.

Input

The first line of the input contains an integer T ($T \leq 10$), indicating the number of cases. For each test case:

The first line: three space separated integers N, M and D ($1 \leq M \leq N, D \leq 100000$). Their meanings are described above.
The following D lines: two space separated integers L and R ($1 \leq L \leq R \leq M$) denoting the consecutive rooms L, L + 1, ..., R which are reversed by the Gru on that day.

Output

For each query, output the number of different ways to arrange minions modulo 880803841 as it can be very large.

Sample Input

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

Sample Output

```
3
1
15
25
15
```

J – Unlucky 89

Description

Amir thinks 89 is an unlucky number for him. But he still wants to know more about 89. He knows 89 is the 24th prime number, and also a Fibonacci number after searching on the internet.
Recently, Amir starts to study some problems about right triangles. He calls a right triangle unlucky if lengths of its both legs are integers and the length of the hypotenuse is a multiple of $\sqrt{89}$. For example, a right triangle with three edges (5,8,$\sqrt{89}$) is unlucky, (4,47,5$\sqrt{89}$), (17,44,5$\sqrt{89}$), and (10,16,2$\sqrt{89}$) are also unlucky. He thinks each unlucky triangle has an energy which is defined by its hypotenuse. The energy of an unlucky triangle is its hypotenuse divided by 89. For example, the energy of (5,8,$\sqrt{89}$) is 1, and the energy of (71,100,13$\sqrt{89}$) is 13. Amir is very curious about the average circumference of all the unlucky triangles whose energy is less than or equal to n. The circumference of a triangle is the sum of its three sides. Can you help him? Amir is expecting a precise answer.

Input

There are multiple cases. Each case is an integer number n in a line. ($1 \leq n \leq 10^8$, the sum of n in the whole input is less than 2×10^8).

Output

Each output for one case has 5 lines. Each output should be in the following format:

```
************************************************
*          Numerator1              Numerator2    __*
*Integer1------------ + Integer2------------ x V89*
*        Denominator1            Denominator2      *
************************************************
```

Each mixed fraction should be irreducible. When numerator is 0, the proper fraction part should be omitted. If Numerator2 is 0 and Integer2 is 1, then "1 x " part should be omitted too. The numerator and the denominator need to be right-aligned. The "—" part should be as same long as the corresponding denominator.
Note the characters other than digits used in the output: '*'(Asterisk, ascii code 42), ' '(Space, ascii code 32), '_'(Underscore, ascii code 95), '-'(Hyphen or minus sign, ascii code 45), 'x'(letter x, ascii code 120), and 'V'(letter V, ascii code 86). Print a blank line after each case. Please refer to the sample output.

Hint

If n is 10^8, there are 637149947 kinds of right triangles in different shapes and sizes.
Explanation for $n = 8$: we can find 10 unlucky right triangles whose energy is no more than 8. They are $(5,8,\sqrt{89})$, $(10,16,2\sqrt{89})$, $(15,24,3\sqrt{89})$, $(20,32,4\sqrt{89})$, $(4,47,5\sqrt{89})$, $(17,44,5\sqrt{89})$, $(25,40,5\sqrt{89})$, $(30,48,6\sqrt{89})$, $(35,56,7\sqrt{89})$, and $(40,64,8\sqrt{89})$. The sum of circumferences is $580+46\sqrt{89}$, so the average circumference is $(580+46\sqrt{89})/10$.

Sample Input

```
1
8
9
89
99999999
```

Sample Output

```
**********
*      __*
*13 + V89*
*        *
**********

***************
*      3    __*
*58 + 4- x V89*
*      5      *
***************

*****************
*   4         __*
*63-- + 5 x V89*
*  11           *
*****************

**********************
*   37     13     __*
*598-- + 48-- x V89*
*   90     18      *
**********************

***********************************************
*          93224754             97600661     __*
*620114538--------- + 51235071--------- x V89*
*         106191655            318574965      *
***********************************************
```