A-Geometry Problem

Tired of all the geometry problems in programming contests, you decided to run away from programming contests and get into game development — without realizing that there were more geometry problems waiting for you! You thought up of a nice two player 2-D arcade game which works as follows — the first player throws a triangular metal piece in a particular direction. The second player then throws another triangular metal piece with the aim of hitting the first player's piece. You're very confident that the game will be a worldwide hit — except that you don't know how to decide whether the two pieces will collide.

You've decided to get rid of your mental block and solve this problem at any cost. At the instant that the second player throws his piece, you store the position and velocity of both the pieces, and you refer to this time as 0. Your game does not allow the pieces to already be in collision at this time. You now want to compute the time at which the two pieces will collide.

Input

The first line contains T, the number of test cases. Each test case contains two lines. The first line describes the first player's piece and the second line describes the second player's piece. Each description contains 8 space separated integers — $x_1, y_1, x_2, y_2, x_3, y_3, v_x, v_y$. (x_i, y_i) denotes the position of one of the i-th vertex of the piece. v_x and v_y denote respectively the x and y components of the velocity of the piece.

Constraints:

- $1 \leq T \leq 100$
- $-10000000 \leq x_1, y_1, x_2, y_2, x_3, y_3, v_x, v_y \leq +10000000$

Output

For each test case, output a single line containing the earliest time at which the two pieces collide. The answer must have a relative/absolute accuracy of 10^{-9}. If the pieces will never collide, output a single line saying '`NO COLLISION`' (quotes for clarity).

Sample Input

```
2
0 0 1 0 0 -1 1 0
2 0 3 0 2 -1 0 0
0 0 1 0 0 -1 5 0
2 0 3 0 2 -1 5 0
```

Sample Output

```
1.000000000
NO COLLISION
```

B-Interleaved Periodic String

An interleaved periodic string S can be written down using the following procedure:

1. Write down any two strings s_1 and s_2 of lengths p_1 and p_2 respectively. The strings must consist of only 0s and 1s, and can possibly be empty.
2. Concatenate some copies of the string s_1 to obtain string S_1.
3. Concatenate some copies of the string s_2 to obtain string S_2.
4. Interleave the strings S_1 and S_2 to obtain S.

To interleave two strings, merge their characters arbitrarily, maintaining the relative order in which they occur in both strings. For example, the strings "101" and "011" can be interleaved to get "011011" or "101011", however they cannot be interleaved to form "110110". Given S, find the minimum possible value of $(p_1 + p_2)$.

Input

The input consists of multiple test cases. The first line contains the number of test cases T. Each of the next T lines contain a string S consisting of only '`0`'s and '`1`'s.

Constraints:

- $1 \leq T \leq 20$
- $1 \leq$ length of $S \leq 16$

Output

Output T lines, one corresponding to each test case, containing the minimum value of $(p_1 + p_2)$ for the corresponding test case.

Sample Input

```
1
0101
```

Sample Output

```
2
```

C-Lap time in a racing circuit

NK, India's lead racing driver, inspected the new race track being prepared for India's first F1 race. In his notebook, he represented the race track as consisting of a series of straight lines joined at vertices. Each straight line represents a straight piece of the track between successive corners and he noted down these distances. Each vertex represents a corner and based on the grip of his team's car, he noted down the maximum speed at which each corner could be taken. At speeds less than or equal to this speed, when the driver turns the steering wheel, the direction of velocity changes instantaneously without any loss in magnitude.

The track is cyclic and has N corners, numbered 0 to $N-1$. C_i represents the maximum speed at which corner i can be taken. The circuit also has N straight line segments, numbered 0 to $N-1$, between these corners. S_i represents the distance between corner i and corner $i+1$. S_{N-1} represents the distance between corner $N-1$ and corner 0. Given the maximum acceleration (A) and maximum deceleration under braking (B) of the car, NK wants to know what is the fastest lap time possible if the car starts at rest at corner 0. Lap time is measured as the time taken to once again cross corner 0. When corner 0 is crossed at the end of the lap, the speed of the car should be low enough that it can be brought to a halt safely by travelling along the circuit while braking continuously. (NK's team is not so cash filled that they can crash a car every lap!)

Input

The first line contains a number T, the number of test cases. The first line of each test case contains three space separated numbers, N, A and B. The next line contains N space separated integers representing $C_0 \dots C_{N-1}$. The next line contains N space separated integers representing $S_0 \dots S_{N-1}$.

Output

For each test case output a single line containing the fastest lap time possible. The answer must have a relative/absolute error of less than 10^{-6}.

Constraints:

- $1 \leq T \leq 100$
- $2 \leq N \leq 100$
- $0 \leq C_i \leq 100$ (unit: $metre/sec^{-1}$)
- $1 \leq S_i \leq 1000$ (unit: $metre$)
- $1 \leq A, B \leq 20$ (unit: $metre/sec^{-2}$)

Notes:

Do not bother about the values for S_i not forming a closed polygon. The track can curve and cross itself in many ways and NK only uses straight lines as approximations.

1. Some useful equations of motion:

   $v = u + at$

   $S = ut + 0.5at^2$

   $v^2 = u^2 + 2aS$

   u = initial speed,

   v = final speed,

   t = duration of acceleration,

   a = magnitude of acceleration,

   S = distance covered

Sample Input

```
1
3 10 10
0 0 0
10 10 10
```

Sample Output

```
6.0
```

D-Lattice Squares

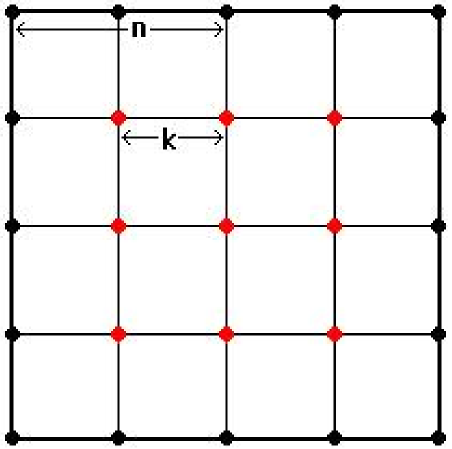
Count the number of distinct squares you can draw using only integer co-ordinates for its 4 corners with the following restrictions.

- The edges should be parallel to the x and y axes
- The x and y co-ordinate of each corner should be within the range 0 to $2 * n$ (range includes 0 and $2 * n$).
- None of the corners should lie in the central $2 * k$ by $2 * k$ square. It means both x and y co-ordinates of the corners should not lie in the range $(n - k)$ to $(n + k)$ (range inclusive of both ends) at the same time.

Two squares are distinct if and only if at least one of its corners is different.

Note that the edges can go through the central forbidden square. The only condition is that the corners itself should not lie in the central forbidden square.

The figure on the right shows the case $n = 2$ and $k = 1$. In the figure the central 2×2 square is forbidden and the forbidden lattice points are marked with red. When drawing a square you cannot use these points (marked red) for any of the corners. You are allowed to use only the points marked with black. So the only allowed square you can draw is the 4×4 square. Hence the answer for this case is 1.



Input

The first line contains one integer t, the number of testcases. $(1 \leq t \leq 50)$

This will be followed by t test cases. Each case is specified in a separate line containing two space separated integers n and k.

Constraints:

- The numbers n and k will be between 1 and 500.
- k will be strictly less than n $(k < n)$.

Output

For each testcase print the number of distinct squares you can draw under the given constraints.

Sample Input

```
2   2 1   3 1
```

Sample Output

```
1   30
```

E-Trip Compulsion

Being a very quirky person, you've modeled your large neighbourhood as a set of numbered junctions and two-way roads connecting these junctions. Somewhat disturbingly, you've actually measured the length of each road in nanometers! Whenever you take a trip from one junction to another, you always note down the lengths of the longest road and the shortest road in your trip and then compute the difference between the two. One day, before you set out on a trip, you're overwhelmed by a strong desire to find out what the lowest possible difference is among all trips that have the same starting and ending junction as yours. Of course, computing all this on paper will take you ages and your trip is a little urgent (you must leave in the next 5 hours), so you decide to write a program.

Input

The first line contains T, the number of test cases. The first line of each test case contains two space separated numbers — N (the number of junctions) and R (the number of roads). The second line of each test case contains the two space separated numbers — *start* (the starting junction of your trip) and *end* (the ending junction of your trip). Each of the next R lines contains three space separated numbers — $from$ (the starting junction of a road), *to* (the ending junction of a road) and *length* (the length of the two-way road connecting $from$ and *to*).

Output

For each test case, output a single line containing one integer — the lowest possible difference. If no trip is possible between *start* and *end*, output a single line saying '`NO PATH`' (quotes for clarity).

Constraints:

- $1 \leq T \leq 10$
- $2 \leq N \leq 2000$
- $0 \leq R \leq 3000$
- $0 \leq start, end, from, to < N$
- $1 \leq length \leq 2,000,000,000$
- *start* and *end* will be different.

Sample Input

```
3
3 2
0 2
0 1 1000
1 2 5000
2 1
0 1
0 1 1000
2 0
0 1
```

Sample Output

```
4000
0
NO PATH
```

F-Visible Lattice

Consider a $N \times N \times N$ lattice. One corner is at (0,0,0) and the opposite one is at (N, N, N). How many lattice points are visible from corner at (0,0,0)? A point X is visible from point Y iff no other lattice point lies on the segment joining X and Y.

Input

The first line contains the number of test cases T. The next T lines contain an interger N.

Output

Output T lines, one corresponding to each test case.

Constraints:

- $T \leq 100$, $1 \leq N \leq 100$

Sample Input

```
3
1
2
5
```

Sample Output

```
7
19
175
```

G-XOR Sum

Given an array of N numbers, we wish to choose a contiguous sub-sequence of the array, so that the bitwise XOR of all chosen numbers is maximum. Bitwise XOR is defined as follows: every bit in the answer is obtained by applying XOR logic on the corresponding bits of the set of numbers. For example 7, 8 and 5 are XORed as follows,

```
Numbers in binary:  0111  1000  0101  -----  1010
```

So the answer is 10 (in decimal). The same answer can be obtained in C/C++/Java by using the XOR operator as 7^8^5.

Input

The first line contains the number of test cases T. The first line of each test-case contains one integer, N (size of the array). The next N lines of each test-case contain integers denoting the elements of the array.

Output

For each test case, output a single line containing the maximum sum that can be obtained.

Constraints:

- $1 \leq T \leq 10$
- $1 \leq N \leq 100,000$
- All input integers will be non-negative and fit into 32 bit signed integer.

Sample Input

```
2  5  3  7  7  7  0  5  3  8  2  6  4
```

Sample Output

```
7  15
```

H-Find The Number

ABC University has k departments. Each department is assigned a department number. There are many students in each department. The university assigns roll numbers to each student such that the number is divisible by the department number. For example if department number is 5, the students can only get roll numbers 5, 10, 15, etc. The purpose is to identify the department of a student easily from his roll number. So if a number is divisible by more than one department number, then that number will not be assigned to any student (so that there will not be any ambiguity). For example if we have departments 5 and 7, then 35, 70, 105, etc are not used because they are divisible by both numbers.

Everything was going fine until one day, someone hacked into the University database and erased the roll number column in the students table! The Database administrator knows that,

- All valid roll numbers (the valid roll numbers are numbers divisible by one and only one department number) less than 1015 were there in the Database.
- All the records were sorted by roll number before the hacker erased them, and the hacking did not change the order of records

Now given the position (1 based index) of the record in the database, can you find out the roll number corresponding to that record quickly?

Input

The first line contains one integer t, the number of testcases ($1 \leq t \leq 50$).

This will be followed by t test cases, each containing 2 lines.

- The first line of each test case gives two numbers k and n separated by space.
- The second line contains k space separated integers specifying the department numbers of each department.

Output

For each test case print the roll number corresponding to the nth record in the database. Output of each test-case should be on a separate line.

Constraints:

- $1 \leq k \leq 12$
- The department numbers will be between 2 and 10^5 (range inclusive of both ends).
- n will fit into a 32 bit signed integer.
- The input will be such that, the answer will always be less than 1015.
- There will not be two departments with same number.
- One department number will not be divisible by another.

Note:

In the first test-case the roll number sequence will be 2, 3, 4, 8, etc (Note that 6 is not a valid roll number). So the 4th number in this sequence is 8.

Sample Input

```
 2   2 4   2 3   3 10   6 11 20
```

Sample Output

```
8   36
```

I-Love for Pizza

My brother and I love pizza. My brother ordered a pizza today with a number of toppings. Some of those toppings I love, like mushrooms, while there are some others that I hate, like olives. Even among the toppings I like (or the ones that I don't like), I like some more than the other, depending on the amount.

Now my brother will let me take a wedge of any size from the pizza. This means I am allowed to make two cuts from the center of the pizza to its circumference, and can keep one of the two resulting pieces. If either cut goes through a topping, the entire topping belongs to that piece which contains the centre of the topping. I am not allowed to cut exactly through the centre of a topping. Each topping will thus remain entirely on one of the pieces. I would like to cut and choose the best piece possible for myself.

Input

Input contains multiple test-cases. The first line of the input contains T, the number of test cases, followed by T testcases. The first line of each test case contains one integer N, the number of toppings. It is followed by N lines containing three space-separated integers each. Each line described a single topping. The first integer denotes my $preference$ for the topping. The next two integers denote respectively the x and y co-ordinates of the centre of the topping.

Output

Output a single line per test-case, indicating the sum of the preferences of all the toppings on the best piece. The best piece is the one that has the maximum sum possible.

Constraints:

- $1 \leq T \leq 25$
- $1 \leq N \leq 10^5$
- $-10^5 \leq preference \leq 10^5$
- The point (x, y) will lie within the pizza, which is assumed to be a circle centered at $(0, 0)$ with a radius of 10^9. The point will not be the centre itself.
- Multiple toppings may be centred at the same point.
- A set of test cases will not exceed 4MB in file size.

Sample Input

```
3
2
-100 28335 972
200 16646 1307
3
7265 341 160
-1000 17646 24060
2735 26741 7225
4
-8609 7286 1522
9243 30219 184
7255 19082 16933
5317 6845 0
```

Sample Output

```
200
10000
21815
```

J-Succession

Vito was involved in a lot of businesses in different areas of New York — from simple olive oil to more dangerous products. Competition was literally cut-throat, and Vito was at his most vulnerable while travelling. So he decided that some of the roads that he travelled on had to be "sanitized" so that he could travel between any two of his areas using only sanitized roads. Since sanitization was an extremely costly process, his Consigliere decided to sanitize the minimum number of roads needed.

All was fine until Vito grew old and decided to hand the reins over to his son Michael. However, his Capos weren't too happy about this as they wanted a part of the business too. So it was decided that Michael would get to pick exactly K areas for himself while the Capos would keep the rest. Michael worked out the business value of each area (including some loss-making areas). He now wants to pick his K areas such that the total business value is maximized and he can travel between any two of his areas using only sanitized roads. Of course, during his travel he does not want to go through an area that is not his. You've been Michael's associate for a long time and you see your chance to impress him and become a full member by telling him what the highest possible value is. Just to prove you're no fluke, you also want to tell him exactly how many ways there are of achieving this. (No big numbers for the Boss, so you will only tell him the remainder this number leaves when divided by 1000000007).

Input

The first line contains T, the number of test cases. The first line of each test case contains three space separated numbers N (the number of areas Vito had), K (the number of areas Michael must choose) and R (the number of sanitized roads). The next line contains N integers, where the ith integer is the business value of the ith area. Each of the next R lines contains two space separated numbers — $from$ and *to* (both 0-based). It implies that there is a two-way sanitized road between area $from$ and area *to*.

Output

Output T lines, one corresponding to each test. On every line, output two space separated numbers. The first number is the highest possible business value. The second number is the number of ways in which this can be done.

Constraints:

- $1 \le T \le 100$
- $1 \le K \le N \le 100$
- $-1000 \le$ value of any business area ≤ 1000
- $0 \le from, to < N$
- No two roads will connect the same pair of areas. $from$ and *to* will be distinct.
- The R roads will be such that it is possible to travel between every pair of areas using only sanitized roads, and removing even one of the roads will make it impossible to travel between every pair of areas using only sanitized roads.

Sample Input

```
1
2 1 1
10 10
0 1
```

Sample Output

```
10 2
```