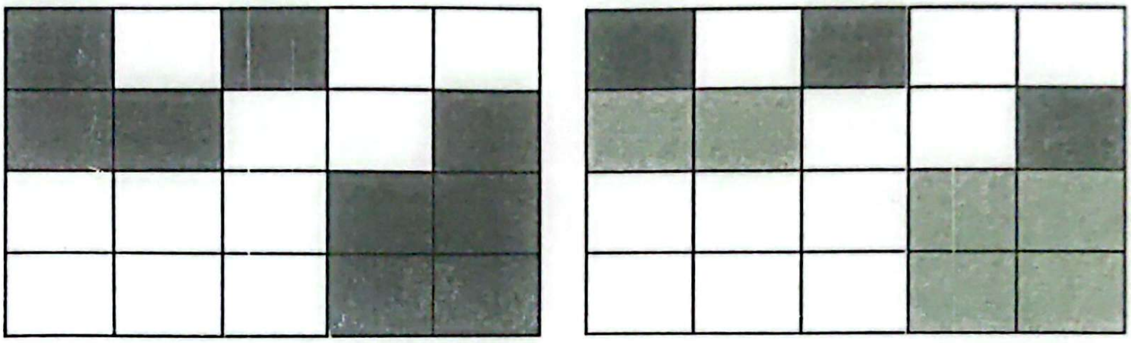
-1 3

Problem A: Aesthetic Islands

Problem

Given an **m x n** 2D grid which represents a map of '1's (land) and '0's (water). An island is surrounded by water and is formed by connecting adjacent lands horizontally, vertically, or diagonally. You may assume all four edges of the grid are all surrounded by water.

You are bored of all the irregular shapes and only want to see how you can shape all of the islands in a rectangle or a box that too while utilizing the maximum space available on any island. So, given a grid of 1s and 0s, determine the number of islands in the grid and the maximum area that can be utilized to make an island look aesthetic.



So, there are 2 islands and the maximum aesthetic areas are 4 and 2.
It is guaranteed that there will be at least one island in the grid.

Input

The first line contains 2 integers **m** and **n**, the dimensions of the grid.
The following m lines contain the grid.

0 0
-2 -1

0 0
4
1
-2 -1 1

Output

An integer on the first line, the number of islands.
Aesthetic areas of all islands in descending order on a single line (space-separated).

Constraints

1 <= n <= 100
1 <= m <= 100

Sample Input	Sample Output

4 5
10100
11001
00011
00011
	2
	4 2

5 6
111000
101000
111011
000011
010111
	3
	6 3 1

Problem A: Kth Bit in Nth Binary String

Problem

Given two positive integers **n** and **k**, the binary string S_n is formed as follows:

$$S_1 = \text{"0"}$$
$$S_i = S_{i-1} + \text{"1"} + \text{reverse(invert(}S_{i-1}\text{))} \text{ for } i > 1$$

Where + denotes the concatenation operation, reverse(x) returns the reversed string x, and invert(x) inverts all the bits in x (0 changes to 1 and 1 changes to 0). For example, the first four strings in the above sequence are:

$S_1 = 0$ $S_2 = 011$ $S_3 = 0111001$ $S_4 = 011100110110001$

Return the k^{th} bit in S_n. It is guaranteed that **k** is valid for the given **n**.

Input

The first line contains an integer **T**, i.e., the number of test cases.
Next **T** lines will contain 2 integers **n** and **k**.

Output

Print a single integer corresponding to each test case.

Constraints

1 <= T <= 1000
1 <= n <= 20
1 <= k <= 2^n - 1

Sample Input	Sample Output
1 3 1	0

2
2 3
4 11
	1
	1

Problem B: Extreme Prime Summation

Problem

The sum of the primes below **10** is **2 + 3 + 5 + 7 = 17**.

Find the sum of all the primes not ~~greater than~~ given **N**.

10

5

1 — 10

Input

First line contains **T** which denotes the number of test cases.
This is followed by **T** lines, each containing an integer, **N**.

3

Output

Print a single integer corresponding to each test case.

2, 5

2

Constraints

$1 <= T <= 10^4$
$1 <= N <= 10^6$

Sample Input	Sample Output
2 5 10	10 17
4 4 8 11 13	5 17 28 41

Al

C A:

Problem B: Drone Swarm

Problem

UET Lahore recently organized an autonomous drone show with a great number of attendees. But you being a tech enthusiast were more interested in drone control rather than the show itself and upon closer inspection, you noticed that the drones were not truly autonomous but in fact were programmed to move in a straight line of their own. Upon further inspection for a minute or 2, you could easily tell which drone was programmed to move in which direction in 3d space. Let's take that time as t = 0.

Let M(t) be the location of the center of the mass of the N drones, and d(t) be the distance between your positio ı and M(t) at time t.

Given your position (x, y, z) and the initial positions of all of the drones along with their associated velocities, determine the minimum value of d(t) at the earliest time. We are only interested in time **1 <= t <= 1000**.

You can assume that drones may pass freely through all of space, including each other and you. All drones have the same mass. **To avoid floating point precision scale "up" the final minimum distance to the nearest integer.**

The center of mass (x_c, y_c, z_c) of N points will be calculated as:
$x_c = (x_1 + x_2 + \ldots + x_N) / N$
$y_c = (y_1 + y_2 + \ldots + y_N) / N$
$z_c = (z_1 + z_2 + \ldots + z_N) / N$

M(t)

Input

The first line contains an integer **T**, the number of test cases.
Following T test cases follow the following format:
The first line of each test case contains 3 integers **x, y, z**, your position on the 3d space.
The second line of each test case contains **N**, the number of drones, followed by **N** lines of the form:
a b c va vb vc, where (a, b, c) is the initial position of the drone at time t = 0, and (va, vb, vc) is the associated velocity.

Output

(0, 0) (-2, -1)

Print 2 integers (space-separated) on a single line, d_{min} (scaled-up to the nearest integer) and t_{min}.

Constraints

1 <= T <= 50
3 <= N <= 100

Sample Input	**Sample Output**

1
0 0 0
4
1 2 3 1 2 3
3 2 1 3 2 1
1 0 0 0 0 -1
0 10 0 0 -10 -1
	4 1

1
1 2 3
3
-5 0 0 1 0 0
-7 0 0 1 0 0
-6 3 0 1 0 0
	4 7

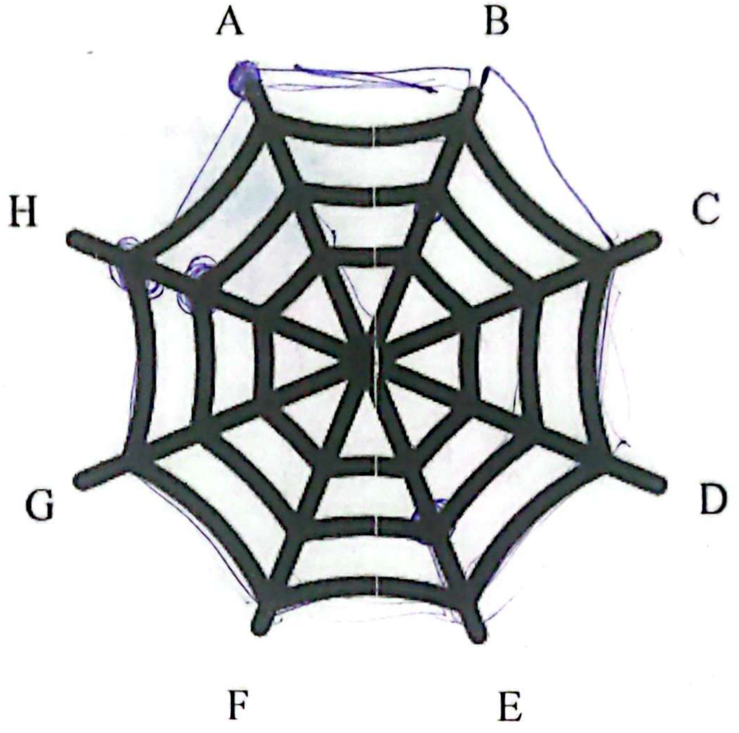
2
4
.
1

-6 1 0
~~-5 1 0~~

Problem C: Robo Spider

Problem

Researchers at UET Lahore created a special robot that moves and behaves like a spider. The track of the robot is also in the shape of a spider web with radials labeled clockwise from the top as **A-H**. The researchers claim that the robot goes from one point of the track to the other in the shortest route possible but you are a little skeptical about that claim and want to test the authenticity. For the given scenario, write a piece of code that takes in coordinates of the starting position of the robot and the goal and prints the minimum cost from start to the end. The cost of every edge is the same. **The track does not go on forever and contains only 100 rings around the center.** The midpoint of the track is labeled **A0.**



Input

The first line contains **T** which denotes the number of test cases.
The first line of the test case contains the starting position of the robot.
The second line of the test case contains the position of the goal.

Output

Print a single integer corresponding to each test case.

Constraints

1 <= T <= 10

Sample Input	**Sample Output**
2 H3 E2 ~~A4~~ B2	4 3
1 A0 G10	10

ar {A:0, B:1, C:2,
D:3, E:4, F:5
G:6, H:7}

H3
E2

s1 A H 3
s2 E 2

int(ar[s2[0]])

Problem D: Manhattan Lamp

Problem

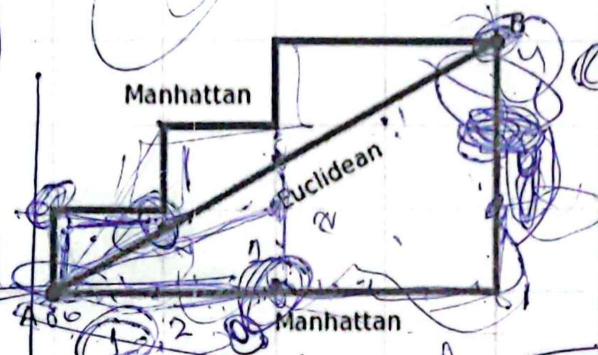
A team of scientists at UET Lahore was working on creating a special type of lamp. While normal lamps emit light in a circle of some Euclidean radius, this special lamp emits light in a circle (not exactly a circle but sort of a square) of Manhattan radius. Like normal lamps, the intensity of light of this lamp also decreases as we go away from the lamp. Let's say that the intensity of light of the lamp at the boundary is 1 and increases by 1 as we move 1 Manhattan distance closer to the lamp. There are some special objects placed there as well that only light up when the light approaching them has a certain intensity, let's call it their rated intensity.

Given the location of the lamp in the 2d space, and the rated intensities of all the objects, determine how many objects will light up, and what will be the minimum Euclidean distance of the nearest lit object. Print -1 if there is no lit object.

To avoid floating point precision scale "up" the final minimum distance to the nearest integer.

(0, 1)

6



6	0	0	1	0	0	0
0	0	1	2	1	0	0
0	1	2	3	2	1	0
1	2	3	4	3	2	1
0	1	2	3	2	1	0
0	0	1	2	1	0	0
0	0	0	1	0	0	0

(total - y)

3

Input

1

The first line contains an integer **T**, i.e., the number of test cases.
Following T test cases follow the following format:
The first line of the test case contains 2 integers **x** and **y**, (x, y) the position of the lamp.
The second line contains **r**, the range of the lamp.
The third line contains **N**, the number of objects, followed by **N** lines of the form:
a b i, where (a, b) is the position of the object and i is the rated intensity.

object (a, b)

i = rated intensity.

Output

For each test case print the number of lit objects on the first line and the minimum Euclidean distance on the second line.
Print -1 if no object lit up.

Constraints

1 <= T <= 1000
1 <= N <= 1000
-1000 <= x, y <= 1000
-1000 <= a, y <= 1000

Sample Input	Sample Output

1
0 0
3
1
2 0 1
	1
	2

2 0 0 3 1 2 0 2 0 0 large. 4 3 objects. 2 0 1 ✓ → 2 1 1 2 ✓ 3 0 1 ✓	-1 3 2

t =