BAPC preliminary contest

October 6, 2007

A Word Ladder

A word ladder is a sequence of words, in which two consecutive words differ by exactly one letter. An example of such a ladder (usually arranged vertically, hence the "ladder") would be: *beer*, *brew*, *brow*, *word*, *down*. Note that to get from one word to the next, the letters may be rearranged, and exactly one letter is changed.

For this problem, you will be given a dictionary of distinct words, all of the same length. Your task is to write a program that finds a word ladder of minimal length, such that the first and last word of the ladder have no letters in common.

Input

On the first line an integer t ($1 \leq t \leq 100$): the number of test cases. Then for each test case:

- A line with two space-separated integers n ($2 \leq n \leq 100$) and l ($1 \leq l \leq 20$): the number of words and their length.
- n lines with a word, each consisting of l lowercase letters (`a` - `z`).

Output

For each testcase:

- a single line with the words in a ladder of minimal length, separated by a single space.

It is guaranteed that at least one such ladder can be constructed. If there is more than one, output the one that comes first lexicographically.

Notes

If s and t are strings of equal length and s_i denotes the ith character of s, then s precedes t lexicographically if for some i: $s_i < t_i$ and $s_j = t_j$ for all $j < i$.

Sample in- and output

Input	Output
1 9 3 alt spy sea opt pea ape spa apt ale	ale alt apt opt

B Bonus Word

Lingo is a once popular game show where the contestants have to guess words. In the original version the contestants would have to guess a five-letter word each round.

In between the rounds of regular word guessing, the contestants can win a bonus prize if they can guess a ten-letter word. The ten-letter word is displayed with the letters permuted. Some letters are colored indicating that they are displayed in the right position. Since there are not that many ten-letter words, it happens frequently that the word is actually a compound: a word constructed by concatenating two shorter words. In this problem we assume that the ten-letter word is always of this form.

Given a dictionary and a sequence of ten letters, you must calculate the possible solutions to the ten-letter word game. Two solutions are considered different if they are constructed from different parts, even if their concatenation is the same. This is illustrated by the the second sample case.

Input

On the first line an integer t ($1 \leq t \leq 100$): the number of test cases. Then for each test case:

- One line with an integer n ($1 \leq n \leq 200$): the number of words in the dictionary.
- n lines with a single word in the dictionary. Each word is between 1 and 9 (inclusive) characters long and consists of only lowercase letters.
- One line with an integer q ($1 \leq q \leq 100$): the number of queries.
- q lines with a single query string. Each query is exactly 10 characters long and will consist of uppercase and lowercase letters. Lowercase letters are in the right position and uppercase letters may be in the wrong position.

All words in the dictionary for a single test case are distinct.

Output

For each test case, output for each query:

- One line with an integer s: the number of solutions.
- $min(1000, s)$ lines, each a solution formatted as two dictionary words separated by a hyphen (`-`).

The solutions to a single query must be ordered lexicographically. If the number of solutions exceeds 1000, then only output the first 1000 solutions.

Notes

If s and t are strings of equal length and s_i denotes the ith character of s, then s precedes t lexicographically if for some i: $s_i < t_i$ and $s_j = t_j$ for all $j < i$. In this problem statement, the hyphen precedes all letters lexicographically.

Sample in- and output

Input	Output
2 5 gunner integral relating tail un 4 TAILGUNNER unINTEGRAL UNrelating IMPOSSIBLE 3 aaaa aaaaa aaaaaa 1 AAAAAAAAAA	6 gunner-tail integral-un relating-un tail-gunner un-integral un-relating 2 un-integral un-relating 1 un-relating 0 3 aaaa-aaaaaa aaaaa-aaaaa aaaaaa-aaaa

C Contest

You and your team are participating in a programming contest. After reading all problems, you have estimated for each problem how long it will take you to solve it. Your teammates have done exactly the same. Now you want to divide the problems, so that the total number of solved problems will be maximized.

Your team is very organized and each team member always writes down all the details on paper. Therefore the computer never forms a bottleneck. The only constraint is that for each team member, the total time required to solve the problems assigned to him should not exceed the time left in the contest.

Input

On the first line an integer t ($1 \leq t \leq 100$): the number of test cases. Then for each test case:

- One line with two integers n ($1 \leq n \leq 10$) and m ($1 \leq m \leq 300$). The number of problems and the number of minutes left in the contest.
- Three lines with n integers each. Each line describes the solving times for a different team member. The jth integer on the ith line is denoted by s_{ij}: either the time in minutes it takes the ith person to solve the jth problem ($1 \leq s_{ij} \leq 300$), or -1 if the person cannot solve this problem.

Output

For each test case:

- One line with the maximum number of problems your team can solve.

Sample in- and output

Input	Output
1 10 300 10 60 -1 -1 10 10 10 240 1 30 15 -1 30 -1 60 60 60 300 5 250 20 -1 -1 60 60 90 90 300 2 245	10

D Cycling

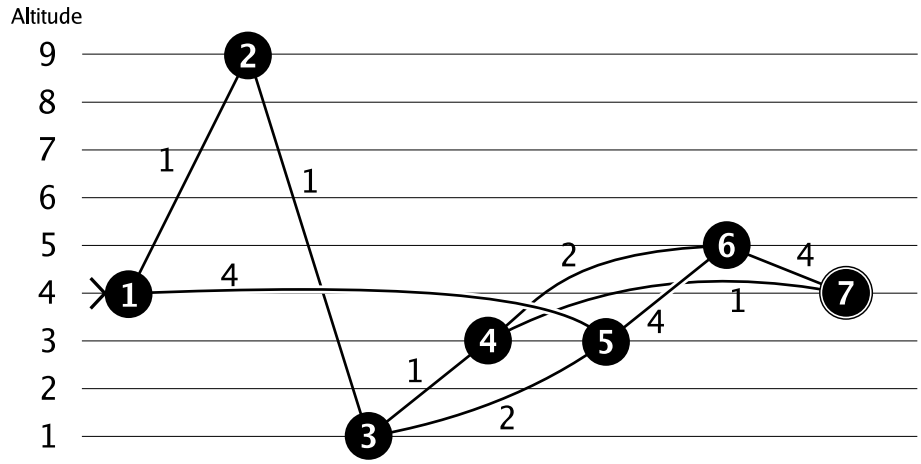
You want to cycle to a programming contest. The shortest route to the contest might be over the tops of some mountains and through some valleys. From past experience you know that you perform badly in programming contests after experiencing large differences in altitude. Therefore you decide to take the route that minimizes the altitude difference, where the altitude difference of a route is the difference between the maximum and the minimum height on the route. Your job is to write a program that finds this route.

You are given:

- the number of crossings and their altitudes, and
- the roads by which these crossings are connected.

Your program must find the route that minimizes the altitude difference between the highest and the lowest point on the route. If there are multiple possibilities, choose the shortest one.

For example:



In this case the shortest path from 1 to 7 would be through 2, 3 and 4, but the altitude difference of that path is 8. So, you prefer to go through 5, 6 and 4 for an altitude difference of 2. (Note that going from 6 directly to 7 directly would have the same difference in altitude, but the path would be longer!)

Input

On the first line an integer t ($1 \leq t \leq 100$): the number of test cases. Then for each test case:

- One line with two integers n ($1 \leq n \leq 100$) and m ($0 \leq m \leq 5\,000$): the number of crossings and the number of roads. The crossings are numbered $1..n$.
- n lines with one integer h_i ($0 \leq h_i \leq 1\,000\,000\,000$): the altitude of the i-th crossing.
- m lines with three integers a_j, b_j ($1 \leq a_j, b_j \leq n$) and c_j ($1 \leq c_j \leq 1\,000\,000$): this indicates that there is a two-way road between crossings a_j and b_j of length c_j. You may assume that the altitude on a road between two crossings changes linearly.

You start at crossing 1 and the contest is at crossing n. It is guaranteed that it is possible to reach the programming contest from your home.

Output

For each testcase, output one line with two integers separated by a single space:

- the minimum altitude difference, and
- the length of shortest path with this altitude difference.

Sample in- and output

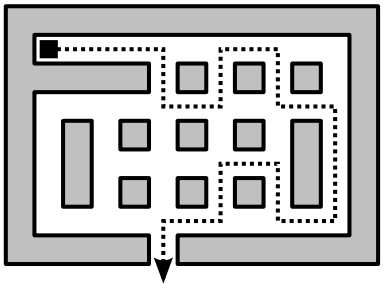
Input	Output
1 7 9 4 9 1 3 3 5 4 1 2 1 2 3 1 3 4 1 4 7 1 1 5 4 5 6 4 6 7 4 5 3 2 6 4 2	2 11

E Escape

You find yourself trapped in a large rectangular room, made up of large square tiles; some are accessible, others are blocked by obstacles or walls. With a single step, you can move from one tile to another tile if it is horizontally or vertically adjacent (i.e. you cannot move diagonally).

To shake off any people following you, you do not want to move in a straight line. In fact, you want to take a turn at every opportunity, never moving in any single direction longer than strictly necessary. This means that if, for example, you enter a tile from the south, you will turn either left or right, leaving to the west or the east. Only if both directions are blocked, will you move on straight ahead. You never turn around and go back!

Given a map of the room and your starting location, figure out how long it will take you to escape (that is: reach the edge of the room).



Input

On the first line an integer t ($1 \leq t \leq 100$): the number of test cases. Then for each test case:

- a line with two integers separated by a space, h and w ($1 \leq h, w \leq 80$), the height and width of the room;
- then h lines, each containing w characters, describing the room. Each character is one of `.` (period; an accessible space), `#` (a blocked space) or `@` (your starting location). There will be exactly one `@` character in each room description.

Output

For each test case:

- A line with an integer: the minimal number of steps necessary to reach the edge of the room, or -1 if no escape is possible.

Sample in- and output

Input	Output
2 9 13 ############# #@.........# #####.#.#.#.# #...........# #.#.#.#.#.#.# #.#.......#.# #.#.#.#.#.#.# #...........# #####.####### 4 6 #.#### #.#.## #...@# ######	31 -1

F Settling Salesman Problem

After travelling around for years, Salesman John has decided to settle. He wants to build a new house close to his customers, so he doesn't have to travel as much any more. Luckily John knows the location of all of his customers.

All of the customers' locations are at (distinct) integer coordinates. John's new house should also be built on integer coordinates, which cannot be the same as any of the customers' locations. Since John lives in a large and crowded city, the travelling distance to any customer is the Manhattan distance: $|x - x_i| + |y - y_i|$, where (x, y) and (x_i, y_i) are the coordinates of the new house and a customer respectively.

What is the number of locations where John could settle, so the sum of the distance to all of his customers is as low as posible?

Input

On the first line an integer t ($1 \leq t \leq 100$): the number of test cases. Then for each test case:

- One line with an integer n ($1 \leq n \leq 2\,000$): the number of customers John has.
- n lines with two integers x_i and y_i ($-1\,000\,000\,000 \leq x_i, y_i \leq 1\,000\,000\,000$): the coordinates of the i-th customer.

Output

For each test case:

- Two space-separated integers: the minimum summed distance to all customers, and the number of spots on which John can build his new house to achieve this minimum.

Sample in- and output

Input	Output
2 4 1 -3 0 1 -2 1 1 -1 2 -999888777 1000000000 1000000000 -987654321	10 4 3987543098 3975087573110998514

G Space

During a programming contest, teams can't sit close to each other, because then a team might copy the solution of another team. You are given the locations of the teams and the minimum required Euclidian distance between two teams. You have to find the number of pairs of teams that sit too close to each other.

Input

On the first line an integer t ($1 \leq t \leq 100$): the number of test cases. Then for each test case:

- One line with two integers n ($1 \leq n \leq 100\,000$) and d ($1 \leq d \leq 50$): the number of teams and the minimum distance between two teams.
- n lines with two integers x_i ($0 \leq x_i \leq 1\,000\,000\,000$) and y_i ($0 \leq y_i \leq 1\,000\,000\,000$): the coordinates of the i-th team. No two teams will have the same coordinates.

Output

For each test case:

- One line with the number of pairs of teams that sit too close to each other.

Notes

The Euclidean distance between points (x_1, y_1) and (x_2, y_2) is $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.

Sample in- and output

Input	Output
1 6 3 0 0 0 3 2 1 2 3 3 0 3 1	8

H Herbert

Herbert is a game in which you control a robot on an infinite two-dimensional grid. There are three commands available to you:

- `s`: Go one square forward in the current direction.
- `l`: Turn ninety degrees counterclockwise.
- `r`: Turn ninety degrees clockwise.

After playing this game for a while, you wonder how many squares you can reach within a certain number of moves. Write a program to calculate the answer to this question.

Input

On the first line an integer t ($1 \leq t \leq 100$): the number of test cases. Then for each test case:

- One line with an integer n ($0 \leq n \leq 1\,000\,000\,000$): the maximum number of moves.

Output

For each test case:

- One line with the number of reachable squares.

Sample in- and output

Input	Output
4 0 1 2 3	1 2 5 11