Problem A
Cookie Crumbs

Source file name: `cookie.c`, `cookie.cpp` *or* `cookie.java`

Cookie Monster likes chocolate chip cookies. He especially likes ones with lots of chocolate chips. Sometimes he likes to experiment with new, unusual cookies. The other day, he tried baking cookies in the shape of rectangles, and using rectangular chocolate chips. Unfortunately, since this was his first time attempting to make rectangular cookies, a few things went wrong. He used way too many chocolate chips. He also turned the oven on too high, and all the chocolate melted and leaked out, leaving only the cookie with holes where the chocolate chips used to be. When the chocolate chips melted, the cookie became disconnected into many cookie crumbs. Cookie Monster needs your help to count the crumbs to make sure he has not lost any in the oven.

Input

The first line of input contains the number of cases. Each case is described by several lines: the first line contanin four integers x_1, y_1, x_2, y_2, each between -1000000000 and 1000000000, giving the x- and y-coordinates of two opposite corners of the cookie. The sides of the cookie are parallel to the x and y coordinate axes. The second line contains an integer $0 \leq n \leq 100$, the number of chocolate chips in the cookie. The following n lines each describe one of the chocolate chips using four integers x_1, y_1, x_2, y_2, each between -1000000000 and 1000000000, the x- and y-coordinates of two opposite corners of the chocolate chip. The sides of each chocolate chip are parallel to the x and y coordinate axes. Chocolate chips can overlap, and they can be partially or completely outside the cookie. The cookie and each chocolate chip will have a non-zero area. A chocolate chip is considered to include the points on its perimeter; therefore, crumbs that would meet only at their corners are considered distinct crumbs.

The input must be read from standard input.

Output

For each test case output a single integer, the number of crumbs (disconnected components) that the cookie splits into after the chocolate chips have melted away.

The output must be written to standard output.

Sample input	**Output for the sample input**
1 0 0 100 100 2 0 0 50 50 50 50 100 100	2

Problem B
Meganominoes

Source file name: `meganominoes.c`, `meganominoes.cpp` *or* `meganominoes.java`

A meganomino is a tile with two ends and a number of dots on each end. Each end can have as few as 0 and as many as 1000000000 dots. In this puzzle, you must place two meganominoes next to each other so that one end of one meganomino touches one end of the other meganomino. The number of dots on the touching ends must be the same on both meganominoes. In addition, the sum of the number of dots on the opposite (non-touching) ends of the meganominoes must be equal to a given number i.

Input

There exists several test cases. The first line of the input contains the number of cases. Each case is describes by several lines: the first line contains two integers $0 < n < 10000$ and $0 < m \leq 100$, the number of meganominoes and the number of queries. The next n lines each contain two integers, the number of dots on each end of a meganomino tile. No two meganomino tiles are the same. The next m lines each contain one of the given sums i.

The input must be read from standard input.

Output

For each test case output a total of m lines, one for each given sum i. On each such line, output a single integer, the number of (unordered) pairs of meganomino tiles that satisfy the constraints. The two tiles in a given pair must be distinct (i.e. a single tile cannot be used twice within a single pair).

The output must be written to standard output.

Sample input	**Output for the sample input**
1 3 2 1 5 6 1 1 7 11 13	1 1

Problem C
Buying Gas

Source file name: `buying.c, buying.cpp` *or* `buying.java`

Gasoline is expensive these days. Many people go out of their way to find the cheapest gas. You could write a computer program to help with this. But on the other hand, searching around too much can waste a lot of time. It could even happen that the driver searches so long that he runs out of gas, which is a major nuisance.

Deciding where to buy gas is even more important on a long trip. Such a trip already takes a long time, so you don't want to waste any more time than you have to. And the consequences of running out are all the more bothersome when far away from home.

Your task here is to write a program that finds the optimal places where to buy gas on a long trip. Of course, the answer must ensure that the car never runs out of gas. Furthermore, it must minimize the number of times that the car stops to fill up.

Input

The input contains several test cases. The first line of input contains the number of cases. Each case is described by several lines: the first line contains three integers $0 < n \leq 100$, $0 \leq m \leq 100000$, and $0 \leq d \leq 100000$, the capacity of the gas tank in litres, the number of gas stations along the route, and the total length of the trip in kilometres. The following m lines each contain two integers, the distance in kilometres from the starting point of the trip to the gas station, and the price of gas at the gas station in tenths of a cent per litre. The car begins the trip with a full tank of gas, and uses 0.1 litres of gas for each kilometre travelled.

The input must be read from standard input.

Output

Find the optimal set of gas stations at which to stop to get gas. For each test case output a single integer, the number of gas stations in this set. If it is not possible to make the trip without running out of gas, output the integer -1.

The output must be written to standard output.

Sample input	**Output for the sample input**
1 50 2 1000 500 1293 750 1337	1

Problem D
Subset Sum

Source file name: `subset.c, subset.cpp` *or* `subset.java`

Maia would like to buy exactly 3.141592 litres of milk. But... Her local grocery store does not stock a bag that size! So Maia decides to buy multiple bags. Even so, it might not be possible to buy a total of exactly 3.141592 litres. In that case, she is willing to buy a little bit more if necessary, but she wants to minimize the extra amount. In addition, Maia wants the bags to all be of distinct sizes, because it would be too boring to buy two bags of the same size. Maia painstakingly figures out which bags of milk to buy. But the next day, she wants 2.718281 litres of milk, and she has to figure it all out again. Clearly she needs to write a program to help her.

Input

The input contains several test cases. The first line of input contains the number of cases. Each case is described by several lines: the first line contains two integers $0 \leq n \leq 1000000000$ and $0 < m \leq 20$, the number of microlitres of milk that Maia wants, and the number of sizes of milk that the store sells. The following m lines each contain an integer $0 \leq a \leq 1000000000$, the size of a bag of milk that the store sells (in microlitres).

The input must be read from standard input.

Output

For each test case output a single integer, the minimum total number of microlitres of milk that Maia needs to buy in order to have at least n microlitres. If it is not possible to buy at least n microlitres, output the word `IMPOSSIBLE`.

The output must be written to standard output.

Sample input	**Output for the sample input**
1 5859870 3 3141592 2718281 1000000	5859873

Problem E
Paper Route

Source file name: `paper.c`, `paper.cpp` *or* `paper.java`

As a poor, tuition-ridden student, you've decided to take up a part time job as a paperboy/papergirl. You've just been handed your paper route: a set of addresses (conveniently labelled 1 to N).

Every morning, you start at the newspaper office (which happens to be address number 0). You have to plan a route to deliver a newspaper to every address - and you also want to get to class right after you're done. Conveniently, there are only N roads in your area connecting the addresses, and each of them takes a known time to traverse. Also, you've precalculated the time it takes to get to campus from each address, including the newspaper office (through some combination of biking, busing, or hitching a ride).

How soon can you be done delivering papers and be in your seat at school?

Input

The input contains several test cases. The first line of input contains the number of cases. Each case is described by several lines: First, there will be a single integer N (the number of addresses, $1 \leq N \leq 100000$). Next, there will be $N+1$ lines, each with an integer c_i (starting with $i = 0$, $0 \leq c_i \leq 1000000000$), the time it takes to get from location i to campus. Finally, the input will contain N lines, each with three integers a, b, c ($0 \leq a, b \leq N$, $a \neq b$, $0 \leq c \leq 1000$). Each of these lines describes a road between locations a and b taking c minutes to traverse.

It is guaranteed that you will be able to reach all the addresses. (Remember that location 0 is the newspaper office.)

The input must be read from standard input.

Output

For each test case output a single integer, the minimum time it will take to deliver all the papers and get to class.

The output must be written to standard output.

Sample input	**Output for the sample input**
2 1 3 4 0 1 1 0 2 2	7

Problem F
Party Location

Source file name: `party.c`, `party.cpp` *or* `party.java`

After the programming contest, all of the contestants would like to throw a party. After the party, however, it will be late, and the contestants will be too tired to walk a long way home. In particular, each contestant refuses to come to the party if it is more than 2.5 km from his or her house.

The solution is to hold the party as close to as many of the contestants' houses as possible. This is where you come in: your job is to determine the optimal location for the party, so that as many contestants as possible will be willing to attend it.

We consider the city to be a flat square, 50 km on each side. A contestant can walk directly from the party in a straight line to his or her house (there are no obstacles).

Input

The input contains several test cases separated by a blank line. The first line of input contains the number of cases. Each case consists of a number of lines, each containing two floating point numbers indicating the (x, y) coordinates of the house of one of the contestants. Each coordinate is between 0.0 and 50.0 (km). Each house is at a distinct location. There are at most 200 contestants.

The last line in the input is a blank line.

The input must be read from standard input.

Output

For each test case output consists of a single integer: the maximum number of contestants that can attend the party.

The output must be written to standard output.

Sample input	**Output for the sample input**
2 4.0 4.0 4.0 5.0 5.0 6.0 1.0 20.0 1.0 21.0 1.0 22.0 1.0 25.0 1.0 26.0 4.0 5.0 4.0 4.0 5.0 6.0 1.0 20.0 1.0 25.0 1.0 22.0 1.0 26.0 1.0 21.0	4 4

Problem G
Numbersrebmun

Source file name: `numbers.c`, `numbers.cpp` *or* `numbers.java`

Anna and Bob are starting up a new high-tech company. Of course, one of their key considerations is choosing a good name for the company. Palindromes are cool. (A palindrome is a word that is the same when reversed, like the names of our two entrepreneurs.) They would really the name of their company to be a palindrome. Unfortunately, they cannot think of a nifty company name that is also a palindrome.

Maybe at least the telephone number of their company could be a palindrome. However, they really want their customers to be able to call them, so they want to choose the company name so that, when it is typed using the letters printed on a phone keypad, the result is also their phone number. (On a standard phone keypad, the following keys contain the corresponding letters: 2: ABC, 3: DEF, 4: GHI, 5: JKL, 6: MNO, 7: PQRS, 8: TUV, 9: WXYZ.)

Input

The first line of input contains a single integer, the number of lines to follow. Each following line contains a company name, which is a string of at most 20 letters, which may be either uppercase or lowercase.

The input must be read from standard input.

Output

For each company name, print a single line of output, containing the word `YES` if the phone number is a palindrome, or `NO` if it is not

The output must be written to standard output.

Sample input	**Output for the sample input**
2 ANBOBNA iAmACoolCompany	YES NO

H - You'll be Working on the Railroad

Congratulations! Your county has just won a state grant to install a rail system between the two largest towns in the county — Acmar and Ibmar. This rail system will be installed in sections, each section connecting two different towns in the county, with the first section starting at Acmar and the last ending at Ibmar. The provisions of the grant specify that the state will pay for the two largest sections of the rail system, and the county will pay for the rest (if the rail system consists of only two sections, the state will pay for just the larger section; if the rail system consists of only one section, the state will pay nothing). The state is no fool and will only consider simple paths; that is, paths where you visit a town no more than once. It is your job, as a recently elected county manager, to determine how to build the rail system so that the county pays as little as possible. You have at your disposal estimates for the cost of connecting various pairs of cities in the county, but you're short one very important requirement — the brains to solve this problem. Fortunately, the lackeys in the computing services division will come up with something.

Input

Input will contain multiple test cases. Each case will start with a line containing a single positive integer $n \leq 50$, indicating the number of railway section estimates. (There may not be estimates for tracks between all pairs of towns.) Following this will be n lines each containing one estimate. Each estimate will consist of three integers s e c, where s and e are the starting and ending towns and c is the cost estimate between them. (Acmar will always be town 0 and Ibmar will always be town 1. The remaining towns will be numbered using consecutive numbers.) The costs will be symmetric, i.e., the cost to build a railway section from town s to town e is the same as the cost to go from town e to town s, and costs will always be positive and no greater than 1000. It will always be possible to somehow travel from Acmar to Ibmar by rail using these sections. A value of n = 0 will signal the end of input.

Output

For each test case, output a single line of the form

```
c1 c2 ... cm cost
```

where each ci is a city on the cheapest path and cost is the cost to the county (note c1 will always be 0 and cm will always be 1 and ci and ci+1 are connected on the path). In case of a tie, print the path with the shortest number of sections; if there is still a tie, pick the path that comes first lexicographically.

Sample Input

```
7
0 2 10
0 3 6
2 4 5
3 4 3
3 5 4
4 1 7
5 1 8
0
```

Sample Output

```
0 3 4 1 3
```