Introduction of CSV:

Mini Project 1:

Little Nanu went crazy yesterday and created an arbitrary matrix consisting of 0, 1 or 2. There was no rule observed in forming this matrix. But then he made up some rules himself.

If there are adjacent 1′s, they are said to be connected. Similarly, if there are adjacent 2′s, they are said to be connected. Here adjacent means all the surrounding positions along with the diagonal positions.

Now given a matrix of 0, 1 or 2, do your computation according to the following rules:

1. If there is a path from any position in top row to any position in bottom row consisting only of 1, then print 1.
2. If there is a path from any position in first column to any position in last column consisting only of 2, then print 2.
3. If both Rule 1 & Rule 2 are true, print AMBIGUOUSAMBIGUOUS.
4. If none of Rule 1, Rule 2 or Rule 3 satisfies, print 0.

Input format

First line of input contains a positive integer N, the size of matrix. Then N line follows, each containing N integers which are either 0, 1 or 2.

Output format

Print a single line containing the output according to the rule mentioned. Clearly, the output can only be 0, 1, 2 or AMBIGUOUS.

Input constraint

1<=N<=100

Example

4

0 0 0 0

2 0 1 0

0 2 1 2

0 1 2 0

Here the output is 2, as there is a connected path of 2 from first column to last column.

SAMPLE INPUT	SAMPLE OUTPUT

3
 	
0 0 1
 	
0 1 2
 	
2 2 1
	AMBIGUOUS

Mini Project 2:

1 Little Nanu as usual has not prepared for his exams. So, to ensure that he does not fail in any course, he goes to the library and picks up **N** books for his **N** courses that he has to give exams for. The number of books being too large, he puts them on different tables and goes to a particular table whenever he wants to read that particular book.

After he is done with studying, he has to put all the books back in their place. He finds that there are exactly **N** empty spaces in the book shelf where he can put his **N** books.

Imagine shelves and tables where books are kept, as locations on the x-coordinate axis. Now Little Nanu wants to minimize the time he takes to put books back in shelves. He takes **T** units of time to put a book back in a shelf, where **T** denotes the distance between the book and the shelf it was put in.

Once Little Nanu puts a book into a shelf, he takes **0 unit** time to reach the next book he wants to put into a shelf. Since he is dumb, and in a hurry, help him calculate the minimum time required to put all the books in shelves.

Note: Once a book has been into a shelf, you can't put another book in the same shelf. Also, initially Nanu can start from any book.

Input format:
The first line contains a pair of integers: an integer **N**, which denotes the number of books and empty shelves. Next line contains **N** space separated integers denoting co-ordinates of the tables where books are placed. Next line contains **N** space separated integers denoting co-ordinates of shelves.

Output format:
Print minimal time it will take Little Nanu to put all books back in shelves.

Mini Project 3:
CSV file handling.
What is CSV file?.

CSV stands for "comma-separated values". In CSV file each value is separated by comma.
Here we have following assumption about data in CSV file .
In first row we have columns name that are separated by comma .
Form second row on-ward we contain data corresponding each columns name for Example:

Text:

```
Title1,Title2,Title3
one,two,three
example1,example2,example3
```

Save it as example.csv

Here we call first line as Header .

If you open it in spreadsheet program like Excel . it will look like as follow :

Title1	Title2	Title3
one	two	three
example1	example2	example3

Write implementation of following operation in Either C++ or C .

1. Create: this function except column name as input argument and create csv file .
   Input : Name , roll_no
 Output: create a csv file with Header Name and Roll_no

2 : IsSameStucture: Compare two CSV files whether they are same structure or not.
      by same structure we mean that number column in both file is same . and there
      is one to one mapping of column name.

2. IsDuplicate: Check if Duplicate row data entries exist in CSV file.

3. RemoveDuplicate: Remove Duplicate raw entry in CSV file.

4. Sort: input CSV file name and column name present in CSV file structure
   Sort the row based on the column value of the column name we enter.

5. Union: Input two CSV file name and Column name.
   Union of row data based on column name and save it to another file name file3.

5. Intersection: Input two CSV file name and Column name present in both file.
   Intersection of row data based on column name and save it to another file name file3

Mini Project 4:

Nanu a software engineer from Mumbai just got placed in a software company in New Delhi. As, he have to relocate from Mumbai to Delhi, he decided to buy a house. Yeah price of houses have decreased after Narendra Modi became PM of India. He decided to buy a house in a locality which has burger points at each corner. As the streets are only one way there is a heavy traffic jam and it take time to Nanu to reach burger point. Modi just declared that he will convert some streets connecting these burger points to two way roads. Some conversion cost is associated with each street. It will cost lot of money to convert all streets to two way streets so Modi wanted these conversion in such a way that total cost of conversion is minimized keeping in mind that all burger points are connected with two way roads. Nanu after coming to know about this idea decided to buy a house in a two way street. But not all streets will be two way. So, he ask you to tell him if he should buy house in a particular street or not. As there can be more than one conversion plan, help Nanu to decide if he should buy house in that street or not. You will be given m queries by Nanu. Your task is to tell for **how many queries will the answer be yes**. Return ans in the form of **a/b** where a is the no. of queries for which ans is yes and b is total number of queries. **(gcd(a,b) should be 1)**.

There will be n burger points and k streets joining them.

[**Input format**]
First line of input contain 1 integer t denoting no.of test cases.

Next line of input will contain 2 positive integers n and k where n is the no.of burger points and k is the no.of streets connecting them.

Burger points are numbered 1 to n.

Next k line contain 3 positive integers x,y,cst each denoting street between x and y and cst is the cost associated with that street.
Next line contains 1 integer m denoting number of queries.
Next m line contains 2 integers c,d.

[**Output Format**]
Output answer in the form as specified in ques. For more clarity see sample test case.

[**Constraints**]
1<=t<=15
1<=n<=1000
1<=cst<=100
k=min(1001, (n*(n-1)/2)) - n) + n
1<=q<=k

SAMPLE INPUT	SAMPLE OUTPUT

1
 	
7
 	
10
 	
1 2 5
 	
1 4 3
 	
1 5 12
 	
1 6 5
 	
4 5 1
 	
5 6 2
 	
5 3 1
 	
3 6 16
 	
4 7 1
 	
2 4 1
 	
5
 	
1 6
 	
2 4
 	
3 5
 	
3 6
 	
1 2
	2/5

Mini Project 5:

we will implement a simple “four-function” calculator using stacks and queues. This calculator takes as input a space-delimited infix expression (e.g. 3+4*7), which you will convert to postfix notation and evaluate. There are four (binary) operators your calculator must handle: addition (+), subtraction (-), multiplication (*), and division (/). In addition, your calculator must handle the unary negation operator (also -). The usual order of operations is in effect:

- the unary negation operator -has higher precedence than the binary operators, and is evaluated right-to-left (right-associative)
- + and - have higher precedence than × and /
- all binary operators are evaluated left-to-right (left-associative)

Mini Project 6:

You are given set of letters, and the challenge is to find how many different words can be made from these letters. This problem is designed to take all the fun out of it by automating the process.

Input:

Input will be in two parts. The first part will be a dictionary of less than 1000 lines, containing words to be searched for. Each line will contain one word of up to 7 characters. Each word will be in lower case. The words will be in alphabetical order. The end of the dictionary will be indicated by a line consisting of a single ‘#’character.

After the dictionary there will be data for several word puzzles, each on a separate line. Each puzzle line will have from one to 7 lower case letters, separated by one or more spaces. Your task is to arrange some or all of these letters to form words in the dictionary. The list of puzzles will be terminated by a line consisting of a single ‘#’.

Output:

For each puzzle line in the input, a single line of output should be produced, containing the number of different words in the dictionary that can be formed using the letters in the puzzle line.

Mini Project 7:

Hind is a country with n cities and m weighted undirected roads such that there is **at least one** possible path between any two cities. It is guaranteed that there is at most one road between any two cities. However, due to a difference in opinion on how the national number of the country should be decided, the people in Hind have decided to separate into two countries. You have been assigned the duty to carry out the separation process in a peaceful way.

You have to divide the cities of Hind into two countries A and B such that there is no path from any city in country A to any city in country B. However, there should still be at least one path between any two cities of country A and any two cities of country B using the original roads of Hind. The only operation that you are allowed to perform is to remove **any one** road of Hind and then assign each city to either country A or country B.

Country A wants their national number to be weight of **minimum spanning tree** of all roads in country A whereas Country B wants their national number to be the weight of **maximum spanning tree** of all roads in country B. To make sure that the separation process is peaceful, you have to remove the road in such a way that the absolute difference between the national numbers of the two countries is as low as possible.

Input
The first line of the input contains two space separated integers **n** and **m**, the number of cities in Hind and the number of roads in Hind respectively. The next m line contains three space separated integers u,v and w in each line where w is the weight of the road between u and v respectively.

Output
If there's no way to carry out the separation satisfying the required conditions of the process print -1 otherwise print the minimum absolute difference between the national numbers of the two countries in a single line.

Mini Project 8:

There are N cities in a Kingdom. Since road infrastructure is not so good, some of the cities are not connected via roads. Now the King is interested in upgrading road infrastructure to such an extent that people can use roadways to travel anywhere in the Kingdom (i.e. road travel is possible between each pair of cities).Following King's initiative, minister cabinet has come up with cost matrix C (the road building cost from city i to city j, details are in input format). You need to print the minimum cost incurred to fulfill the initiative. It is always possible to make the kingdom road connected with given plan.

Note: All roads are bidirectional.

Input:
The first line contains **T** – the number of test cases. First line of each test case contains two integers **N** – the number of cities and **M** –the number of existing roads.
Next M lines contain city id pairs **u** and **v** (1 - based indexed) for existing roads between cities.
Next line contains a single integer **K** – the number of additional roads that can be built. K lines follow.
Each of these K lines consists of three space separated integers – **u, v and c** denoting a road between cities u and v with cost c.
1 <= T <= 10
1 <= N <= 10000
1 <= K+M <= 1000000
1 <= C <= 1000000
1 <= u, v <= N

Output:
Print the minimum cost incurred to connect cities in a way such that there exists a path between every two cities.

Mini Project 9:
Write a code to solve linear system of equations.

Input:
Number of Variable: n
Number of Equation: m
Enter value for a[i][j] and c[i]

$1 \le i \le m$ and $1 \le j \le n$
Here a[i][j] , c[i]correspond to following value in equation .

a[1][1] x[1] + a[1][2] x[2] +a[1][3]x[3] ---- a[1][n-1]x[n-1]= c[1]
a[2][1] x[1] + a[2][2] x[2] +a[2][3]x[3] ---- a[2][n-1]x[n-1]= c[2]
………………………………………………………………
………………………………………………………………
a[m][1] x[1] + a[m][2] x[2] +a[m][3]x[3] ---- a[m][n-1]x[n-1]= c[m]

output:
if there is unique solution then print then
print the value of x[1] to x[n-1].
Else
if no solution possible then print “no solution”.

if multiple solution possible then print “multiple solution” .

Let if we have following liner equation :
2x+3y=5
x+y=2

Sample Input:
2 2
2 3 5
1 1 2

Output:
1 1

Mini Project 10:
Sudoku is a Japanese, fun puzzle game. It requires the player to fill in the 9x9 square matrix with numbers one to nine. The numbers should be arranged in such a way that each row, column or box contains one of each number.
Partially filled input in 9x9 square matrix with numbers one to nine.
All other entry is 0. 0 mean cell is empty can be filled with value one to nine.

Input:
Partially filled input in 9x9 square matrix with numbers one to nine.
All other entry is 0. 0 mean cell is empty can be filled with value one to nine.

Output:
9x9 square matrix each cell contain numbers from one to nine.

Sample Input

0	4	6	1	0	5	2	8	0
1	0	2	3	0	7	4	0	0
0	8	9	0	0	7	6	0	0
9	6	1	0	4	0	0	0	0
4	0	8	0	5	0	0	9	2
0	0	0	8	7	0	1	0	0
2	1	0	0	0	6	9	0	7
8	9	0	0	3	0	0	0	0

0	0	0	0	1	4	0	0	5

sample Output

7	4	6	1	9	5	2	8	3
1	5	2	3	6	8	7	4	9
3	8	9	4	2	7	6	5	1
9	6	1	2	4	3	5	7	8
4	7	8	6	5	1	3	9	2
5	2	3	8	7	9	1	6	4
2	1	4	5	8	6	9	3	7
8	9	5	7	3	2	4	1	6
6	3	7	9	1	4	8	2	5

Mini Project 11:

Assume the cricket ground to be an infinite grid and there can be more than 11 players in a team. It is known that the batsmen stand at the point (0,0). The field placement follows the following pattern

1st fielder is to be placed at 1 step to the east of batsman.
2nd fielder is to be placed at 2 steps to the north of 1st fielder
3rd fielder is to be placed at 3 steps to the West of 2nd fielder
4th fielder is to be placed at 4 steps to the south of 3rd fielder
5th fielder is to be placed at 5 steps to the East of 4th fielder
6th fielder is to be placed at 6 steps to the North of 5th fielder

And so on….

The fielders can be placed at only these positions and nowhere else…i.e at (1,0) , (1,2) , (-2,2) , (-2,-2) and so on.

In the ICC Champions Trophy   ind vs pak match in4 jun 2017 , Virat Kohli realizes that to win the match, it is crucial to get Pakistani opener out early. After a deep study, it was found that Pakistani opener often gets out by giving catch to fielders standing at prime numbered positive positions of **x** on **X-Y plane** on the grid. He has decided that he will himself take the catch. Now, Clarke is confused and wants your help to win the match. Find out for him whether Kohli has positioned himself at any of the catching positions or not.

Input :
Input starts with T denoting the number of test cases.
Next T lines contains space separated coordinates x,y which are coordinates of point where Clarke has positioned himself.

Output :
If the position is wicket taking , then print “YES” without quotes else Print “NO”.

Constraints :
1 ≤ T ≤ 100
1 ≤ x,y ≤ 10^5

SAMPLE INPUT	SAMPLE OUTPUT
4	NO
3 3	YES
3 4	NO
5 5	NO
6 6	

Mini 12:

Good news , You have a chance to work in IRCTC as a software developer . Applications are required with a program for railway sites . sid of a station is a something which uniquely defines it . Given N station's sid from 1 to N and distance between any two stations , if any person wants to go from station having sid A to another station C via station B as early as possible , where time is directly proportional to distance . What you have to do is to print the total distance and path to be covered by train if it exists , otherwise print **"No Train Found."**. If a railway line connects station A to B with distance d than it means it will also connects station B to station A with same distance d .

Input
First line of the input contains the number of test cases T. It is followed by T tests cases. Each test case's first line contains two integers N and k , where N is the number of stations having sid from 1 to N and k is the number of railway lines which connects stations. Following k lines each contain three numbers a , b and d which represents station having sid a connects station b and distance between them is d . After that a line follows with number A , B , C , where A represents sid of the source station , C the destination station and B is the station via which person wants to go.

Output
For each test case , If there exists no paths as described in problem print **"No Train Found."** otherwise in first line print the total distance and in second line path to be covered .

Constraints
1 <= t <= 100
1 <= k <= 100000
3 <= N <= 100000
1 <= A,B,C <= N
1 <= d <= 10000000

SAMPLE INPUT	SAMPLE OUTPUT

2
 	
6 6
 	
1 2 2
 	
2 5 5
 	
2 3 4
 	
1 4 1
 	
4 3 3
 	
3 5 1
 	
1 3 6
 	
10 10
 	
1 5 78
 	
1 8 221
 	
2 7 92
	No Train Found.
	
	692
	
	6 3 5 1 8 2

2 8 159
 	
3 5 55
 	
3 6 179
 	
3 10 237
 	
4 8 205
 	
5 6 191
 	
8 10 157
 	
6 3 2	

Mini Project 13:

Little Nanu wants to get a job offer during placement as soon as possible. To increase his chances of getting a job, he wants to give as many interviews at as many different companies as possible. There are **N** rooms, which are labeled from 1 to **N**, and in each room a different company is doing its interview process. Between these **N** rooms, we have **M** paths which guarantee the fact that the Nanu can reach any room from any given room he's in. Every **M** path takes some **T** time to travel between two rooms.

Importantly, he wants to maximize his travel time so he can revise his concepts while traveling. Since he's busy preparing for his interviews, help him maximize his travel time so he can prepare for his interviews. Obviously, there will be no cycle allowed in this graph. Since Nanu is also enlightened, once he has traveled on a particular path, he takes 0 units of time to travel on it again.

Input format:
The first line contains an integer **TC**, which denotes the number of test cases. The next line contains two integers, **N** and **M**, denoting the number of rooms and number of paths. The next **M** lines contain three integers **U**, **V**, and **T**, where **U** and **V** denote the two edges being joined, and **T** denoting the time taken to travel between **U** and **V**.

Output format:
Print the maximum time obtained by Nanu to revise his concepts to each test case in a new line.