Uva 10987 Problem F

AntiFloyd

Time Limit: 2 seconds

"There is nothing new under the sun but
there are lots of old things we don't know."

Ambrose Bierce

You have been hired as a systems administrator for a large company. The company head office has **n** computers connected by a network of **m** cables. Each cable connects two different computers, and there is at most one cable connecting any given pair of computers. Each cable has a latency, measured in micro-seconds, that determines how long it takes for a message to travel along that cable. The network protocol is set up in a smart way, so that when sending a message from computer A to computer B, the message will travel along the path that has the smallest total latency, so that it arrives at B as soon as possible. The cables are bi-directional and have the same latency in both directions.

As your first order of business, you need to determine which computers are connected to each other, and what the latency is along each of the **m** cables. You soon discover that this is a difficult task because the building has many floors, and the cables are hidden inside walls. So here is what you decide to do. You will send a message from every computer A to every other computer B and measure the latency. This will give you **n**(**n**-1)/2 measurements. From this data, you will determine which computers are connected by cables, and what the latency along each cable is. You would like your model to be simple, so you want to use as few cables as possible.

Input

The first line of input gives the number of cases, **N** (at most 20). **N** test cases follow. Each one starts with a line containing **n** (0<**n**<100). The next **n**-1 lines will contain the message latency measurements. Line**i** will contain **i** integers in the range [1, 10000]. Integer **j** is the amount of time it takes to send a message from computer **i**+1 to computer **j** (or back).

Output

For each test case, output a line containing "Case #**x**:". The next line should contain **m** - the number of cables. The next **m** lines should contain 3 integers each: **u**, **v** and **w**, meaning that there is a cable between computers **u** and **v**, and it has latency **w**. Lines should be sorted first by **u**, then by **v**, with **u**<**v**. If there are multiple answers, any one will do. If the situation is impossible, print "Need better measurements." Print an empty line after each test case.

Sample Input	Sample Output
2 3 100 200 100 3 100 300 100	Case #1: 2 1 2 100 2 3 100 Case #2: Need better measurements.

Uva 10801 Problem ?

Lift Hopping

Time Limit: 1 second

> Ted the bellhop: *"I'm coming up and if there isn't a dead body by the time I get there, I'll make one myself. You!"*
>
> Robert Rodriguez, "Four Rooms."

A skyscraper has no more than 100 floors, numbered from 0 to 99. It has **n** (1<=**n**<=5) elevators which travel up and down at (possibly) different speeds. For each **i** in {1, 2,... n}, elevator number **i** takes $\mathbf{T_i}$(1<=$\mathbf{T_i}$<=100) seconds to travel between any two adjacent floors (going up or down). Elevators do not necessarily stop at every floor. What's worse, not every floor is necessarily accessible by an elevator.

You are on floor 0 and would like to get to floor **k** as quickly as possible. Assume that you do not need to wait to board the first elevator you step into and (for simplicity) the operation of switching an elevator on some floor always takes exactly a minute. Of course, both elevators have to stop at that floor. You are forbiden from using the staircase. No one else is in the elevator with you, so you don't have to stop if you don't want to. Calculate the minimum number of seconds required to get from floor 0 to floor **k** (passing floor **k** while inside an elevator that does not stop there does not count as "getting to floor **k**").

Input

The input will consist of a number of test cases. Each test case will begin with two numbers, **n** and **k**, on a line. The next line will contain the numbers $\mathbf{T_1}$, $\mathbf{T_2}$,... $\mathbf{T_n}$. Finally, the next **n** lines will contain sorted lists of integers - the first line will list the floors visited by elevator number 1, the next one will list the floors visited by elevator number 2, etc.

Output

For each test case, output one number on a line by itself - the minimum number of seconds required to get to floor **k** from floor 0. If it is impossible to do, print "IMPOSSIBLE" instead.

Sample Input	Sample Output
2 30 10 5 0 1 3 5 7 9 11 13 15 20 99 4 13 15 19 20 25 30 2 30 10 1 0 5 10 12 14 20 25 30 2 4 6 8 10 12 14 22 25 28 29 3 50 10 50 100 0 10 30 40 0 20 30 0 20 50 1 1 2 0 2 4 6 8 10	275 285 3920 IMPOSSIBLE

Explanation of examples

In the first example, take elevator 1 to floor 13 (130 seconds), wait 60 seconds to switch to elevator 2 and ride it to floor 30 (85 seconds) for a total of 275 seconds.

In the second example, take elevator 1 to floor 10, switch to elevator 2 and ride it until floor 25. There, switch back to elevator 1 and get off at the 30'th floor. The total time is
10*10 + 60 + 15*1 + 60 + 5*10 = 285 seconds.

In example 3, take elevator 1 to floor 30, then elevator 2 to floor 20 and then elevator 3 to floor 50.

In the last example, the one elevator does not stop at floor 1.

Uva 10841 Problem B

Lift Hopping in the Real World

Time Limit: 1 second

"I think problem descriptions should withstand the reality check, otherwise they are misleading."

little joey

A skyscraper has no more than 100 floors, numbered from 0 to 99. It has **n** (1<=**n**<=50) elevators which travel up and down at (possibly) different speeds. For each **i** in {1, 2,... **n**}, elevator number **i** takes$\mathbf{T_i}$ (1<=$\mathbf{T_i}$<=100) seconds to travel between any two adjacent floors (going up or down). Elevators do not necessarily stop on every floor. What's worse, not every floor is necessarily accessible by an elevator.

You are on floor 0 and would like to get to floor **k** as quickly as possible. When you first arrive to floor 0, each of the **n** elevators could be parked on any of the floors accessible by that elevator. You can take any of the elevators that stop on floor 0 and you can switch elevators on any floor, f, if both of them stop on floor f. Assume that it takes 5 seconds to exit one elevator and push another elevator's button. You are forbiden from using the staircase. No one else is in the building, so you don't have to stop if you don't want to, and no one else will request an elevator. Each elevator has its own button, and the system has built-in security to prevent abuse by evil teenagers - once you press the button calling for some elevator E, all the buttons calling for other elevators will be disabled until elevator E arrives at your floor. In other words, you can only call for one elevator at a time.

Calculate the number of seconds required to get from floor 0 to floor **k** in the worst case if you do not know the initial positions of the elevators. Passing floor **k** while inside an elevator that does not stop there does not count as "getting to floor **k**". See the examples below.

Input

The input will consist of a number of test cases. Each test case will begin with two numbers, **n** and **k**, on a line. The next line will contain the

numbers $\mathbf{T_1}$, $\mathbf{T_2}$,... $\mathbf{T_n}$. The next **n** lines will contain sorted lists of integers - the first line will list the floors visited by elevator number 1, the next one - those visited by elevator number 2, etc.

Output

For each test case, output one number on a line - the number of seconds required to get to floor **k** from floor 0 in the worst case. If floor **k** is inaccessible from floor 0, print "IMPOSSIBLE" instead.

Sample Input	Sample Output
2 30 10 5 0 1 3 5 7 9 11 13 15 20 99 4 13 15 19 20 25 30 2 30 10 1 0 5 10 12 14 20 25 30 2 4 6 8 10 12 14 22 25 28 29 3 50 10 50 100 0 10 30 40 0 20 30 0 20 50 1 1 2 0 2 4 6 8 10	1295 600 8505 IMPOSSIBLE

Discussion

In the first example, the worst case is when elevator 1 is on floor 99 and will take 990 seconds to reach floor 0. You will then take it to floor 13 in 130 seconds, spend 5 second exiting and calling elevator 2, which is on floor 30. It will take 170 seconds to reach you and 170 seconds to take you to floor 30. The total is 1295 seconds.

In the second example, the only sensible way to get to floor 30 is to ride the first elevator all the way. Switching to elevator 2 and then back will only make things worse because elevator 1 must take at least 300 seconds to reach floor 30. In the worst case, elevator 1 starts on floor 30.

In example 3, the security system will not allow you to push all 3 buttons and see which elevator comes first. Instead, you should push elevator 2's button. Unfortunately, it is on floor 30 and takes 1500 seconds to get to you. You board it and it takes you to floor 20 in 1000 seconds. You spend 5 seconds pushing the button for elevator 3, which is on floor 50. It takes 3000 seconds before it reaches floor 20 and 3000 more seconds to take you to floor 50. The total is 8505 seconds. If you take elevator 1 first, your worst case time will be 8710 seconds.

In the last example, the one elevator does not stop at floor 1.

POJ 1724 ROADS

Time Limit:1000MS Memory Limit:65536K

Description

N cities named with numbers 1 ... N are connected with one-way roads. Each road has two parameters associated with it : the road length and the toll that needs to be paid for the road (expressed in the number of coins).
Bob and Alice used to live in the city 1. After noticing that Alice was cheating in the card game they liked to play, Bob broke up with her and decided to move away - to the city N. He wants to get there as quickly as possible, but he is short on cash.

We want to help Bob to find **the shortest path** from the city 1 to the city N **that he can afford** with the amount of money he has.

Input

The first line of the input contains the integer K, 0 <= K <= 10000, maximum number of coins that Bob can spend on his way.
The second line contains the integer N, 2 <= N <= 100, the total number of cities.

The third line contains the integer R, 1 <= R <= 10000, the total number of roads.

Each of the following R lines describes one road by specifying integers S, D, L and T separated by single blank characters :

- S is the source city, 1 <= S <= N
- D is the destination city, 1 <= D <= N
- L is the road length, 1 <= L <= 100
- T is the toll (expressed in the number of coins), 0 <= T <=100

Notice that different roads may have the same source and destination cities.

Output

The first and the only line of the output should contain the total length of the shortest path from the city 1 to the city N whose total toll is less than or equal K coins.
If such path does not exist, only number -1 should be written to the output.

Sample Input

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

Sample Output

```
11
```