acm International Collegiate Programming Contest

IBM event sponsor

2014 Carnegie Mellon University

15-295 Competition Programming and Problem Solving

End of Term Party

6:30pm-9:30pm, Dec. 3, 2014

PROBLEMS

A:	Fix
B:	Pollution Solution
C:	Fiber Network
D:	Space Turtle
E:	Hike on a Graph
F:	Raisins
G:	Towers
I:	Keyboard

Hints:

1. Problems are not sorted from easiest to hardest. It is important to solve the easiest problems first to minimize your penalty score. Keep an eye on the scoreboard to find easy problems.
2. Be careful about the efficiency of input/output. If the input size is larger than 10^5 bytes, try to use "`scanf / printf`" in C++ and "`BufferedReader / BufferedWriter`" in Java.
3. Do not access the internet, except for some language support and our online judge for submissions.

Problem A: Fix

A collection of words is *prefix-free* if no word is a prefix of any other word. A collection of words is *suffix-free* if no word is a suffix of any other word. A collection of words is *fix-free* if it is both prefix-free and suffix-free.

For this problem, a word is a sequence of lower-case letters of length between 1 and 25. A word *X* is a prefix of word *Y* if *X* consists of the first *n* characters of *Y*, in order, for some *n*. That is, the word "cat" has prefixes "c", "ca", and "cat". Similarly, a word *X* is a suffix of *Y* if *X* consists of the last *n* characters of *Y*, in order, for some *n*.

Your input will be 3*N*+1 lines: the first line will be the number *N*, and the remaining 3*N* lines will be the *N* collections of 3 words each. (That is, lines 2, 3, and 4 compose the first collection, lines 5, 6, and 7 compose the second collection, and so on).

Your output will be *N* lines, each line containing either Yes (if that collection of words is fix-free) or No (if that collection is not fix-free).

Sample Input
2
abba
aab
bab
a
ab
aa

Sample Output
Yes
No

Pollution Solution

Time Limit: 1 second

As an employee of Aqueous Contaminate Management, you must monitor the pollution that gets dumped (sometimes accidentally, sometimes purposefully) into rivers, lakes and oceans. One of your jobs is to measure the impact of the pollution on various ecosystems in the water such as coral reefs, spawning grounds, and so on.

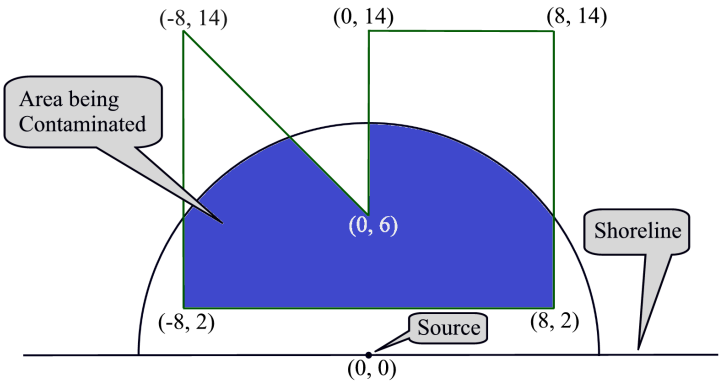


Figure J.1: Illustration of Sample Input 1.

The model you use in your analysis is illustrated in Figure J.1. The shoreline (the horizontal line in the figure) lies on the x-axis with the source of the pollution located at the origin (0,0). The spread of the pollution into the water is represented by the semicircle, and the polygon represents the ecosystem of concern. You must determine the area of the ecosystem that is contaminated, represented by the dark blue region in the figure.

Input

The input consists of a single test case. A test case starts with a line containing two integers n and r, where $3 \leq n \leq 100$ is the number of vertices in the polygon and $1 \leq r \leq 1\,000$ is the radius of the pollution field. This is followed by n lines, each containing two integers x_i, y_i, giving the coordinates of the polygon vertices in counter-clockwise order, where $-1\,500 \leq x_i \leq 1\,500$ and $0 \leq y_i \leq 1\,500$. The polygon does not self-intersect or touch itself. No vertex lies on the circle boundary.

Output

Display the area of the polygon that falls within the semicircle centered at the origin with radius r. Give the result with an absolute error of at most 10^{-3}.

Sample Input 1	Sample Output 1
6 10 -8 2 8 2 8 14 0 14 0 6 -8 14	101.576437872

Problem C: Fiber Network

Several startup companies have decided to build a better Internet, called the "FiberNet". They have already installed many nodes that act as routers all around the world. Unfortunately, they started to quarrel about the connecting lines, and ended up with every company laying its own set of cables between some of the nodes.
Now, service providers, who want to send data from node *A* to node *B* are curious, which company is able to provide the necessary connections. Help the providers by answering their queries.

Input Specification

The input contains several test cases. Each test case starts with the number of nodes of the network *n*. Input is terminated by *n=0*. Otherwise, *1<=n<=200*. Nodes have the numbers *1*, ..., *n*. Then follows a list of connections. Every connection starts with two numbers *A*, *B*. The list of connections is terminated by*A=B=0*.
Otherwise, *1<=A,B<=n*, and they denote the start and the endpoint of the unidirectional connection, respectively. For every connection, the two nodes are followed by the companies that have a connection from node *A* to node *B*. A company is identified by a lower-case letter. The set of companies having a connection is just a word composed of lower-case letters.
After the list of connections, each test case is completed by a list of queries. Each query consists of two numbers *A*, *B*. The list (and with it the test case) is terminated by *A=B=0*. Otherwise, *1<=A,B<=n*, and they denote the start and the endpoint of the query. You may assume that no connection and no query contains identical start and end nodes.

Output Specification

For each query in every test case generate a line containing the identifiers of all the companies, that can route data packages on their own connections from the start node to the end node of the query. If there are no companies, output `"-"` instead. Output a blank line after each test case.

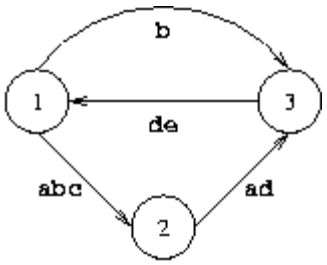
Sample Input

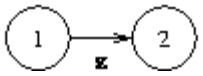
```
3
1 2 abc
2 3 ad
1 3 b
3 1 de
0 0
1 3
2 1
3 2
0 0
2
1 2 z
0 0
1 2
2 1
0 0
0
```

Sample Output

```
ab
d
-

z
-
```





Problem D: Space Turtle

Space Turtle is a fearless space adventurer. His spaceship, the Tortoise, is a little outdated, but still gets him where he needs to go.

The Tortoise can do only two things – move forward an integer number of light-years, and turn in one of four directions (relative to the current orientation): right, left, up and down. In fact, strangely enough, we can even think of the Tortoise as a ship which travels along a 3-dimensional co-ordinate grid, measured in light-years.

In today's adventure, Space Turtle is searching for the fabled Golden Shell, which lies on a deserted planet somewhere in uncharted space. Space Turtle plans to fly around randomly looking for the planet, hoping that his turtle instincts will lead him to the treasure.

You have the lonely job of being the keeper of the fabled Golden Shell. Being lonely, your only hobby is to observe and record how close various treasure seekers come to finding the deserted planet and its hidden treasure. Given your observations of Space Turtle's movements, determine the closest distance Space Turtle comes to reaching the Golden Shell.

Input
The first line consists of three integers sx, sy, and sz, which give the coordinates of Space Turtle's starting point. Space Turtle is originally oriented in the positive x direction, with the top of his spaceship pointing in the positive z direction, and with the positive y direction to his left. Each of these integers are between –100 and 100. The second line consists of three integers tx, ty, and tz, which give the coordinates of the deserted planet. Each of these integers are between –10000 and 10000. The rest of the lines describe Space Turtle's flight plan in his search for the Golden Shell. Each line consists of an integer, d, $0 <= d <= 100$, and a letter c, separated by a space. The integer indicates the distance in light-years that the Tortoise moves forward, and the letter indicates the direction the ship turns after having moved forward. 'L', 'R', 'U', and 'D' stand for left, right, up and down, respectively. There will be no more than 100 such lines.

On the last line of input, instead of one of the four direction letters, the letter 'E' is given instead, indicating the end of today's adventure.

Output
Output the closest distance that Space Turtle gets to the hidden planet, rounded to 2 decimal places. If Space Turtle's coordinates coincide with the planet's coordinates during his flight indicate that with a distance of 0.00. He safely lands on the planet and finds the Golden Shell.

Sample Input
```
0 0 0
1 1 1
2 L
2 L
2 U
2 U
```

2 L
2 L
2 U
2 E

Sample Output
1.41

Problem E: Hike on a Graph

"Hike on a Graph" is a game that is played on a board on which an undirected graph is drawn. The graph is complete and has all loops, i.e. for any two locations there is exactly one arrow between them. The arrows are coloured. There are three players, and each of them has a piece. At the beginning of the game, the three pieces are in fixed locations on the graph. In turn, the players may do a move. A move consists of moving one's own piece along an arrow to a new location on the board. The following constraint is imposed on this: the piece may only be moved along arrows of the same colour as the arrow between the two opponents' pieces.

In the sixties ("make love not war") a one-person variant of the game emerged. In this variant one person moves all the three pieces, not necessarily one after the other, but of course only one at a time. Goal of this game is to get all pieces onto the same location, using as few moves as possible. Find out the smallest number of moves that is necessary to get all three pieces onto the same location, for a given board layout and starting positions.

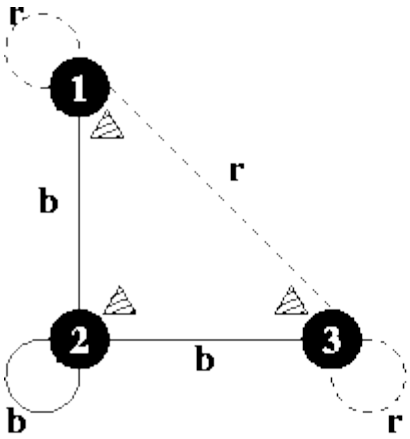
Input Specification

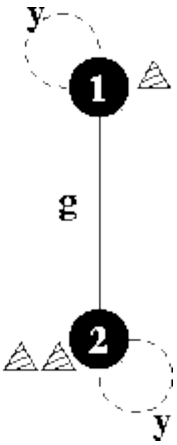
The input file contains several test cases. Each test case starts with the number n. Input is terminated by $n=0$. Otherwise, $1<=n<=50$. Then follow three integers p_1, p_2, p_3 with $1<=p_i<=n$ denoting the starting locations of the game pieces. The colours of the arrows are given next as a $m \times m$ matrix of whitespace-separated lower-case letters. The element m_{ij} denotes the colour of the arrow between the locations i and j. Since the graph is undirected, you can assume the matrix to be symmetrical.

Output Specification

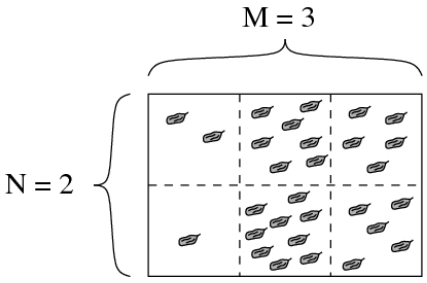
For each test case output on a single line the minimum number of moves required to get all three pieces onto the same location, or the word "impossible" if that is not possible for the given board and starting locations.

Sample Input	Sample Output
3 1 2 3 r b r b b b r b r 2 1 2 2 y g g y 0	2 impossible





Problem F: RAISINS



Plovdiv's famous master chocolatier Bonny needs to cut a slab of chocolate with raisins. The chocolate is a rectangular block of identical square pieces. The pieces are aligned with the edges of the chocolate, and they are arranged in ***N*** rows and ***M*** columns, for a total of ***N*M*** pieces. Each piece has one or more raisins on it, and no raisins lie between or across pieces.
Initially, the chocolate is one single, monolithic block. Bonny needs to cut it into smaller and smaller blocks until finally she has cut the chocolate down to its ***N*M*** individual pieces. As Bonny is very busy, she needs the help of her assistant, Sly Peter, to do the cutting. Peter only makes straight line, end-to-end cuts and he wants to be paid for every single cut he makes. Bonny has no money at hand, but she has plenty of raisins left over, so she offers to pay Peter in raisins. Sly Peter agrees to this arrangement, but under the following condition: every time he cuts a given block of chocolate into two smaller blocks, he has to be paid as many raisins as there are on the block he was given.
Bonny wants to pay Peter as little as possible. She knows how many raisins there are on each of the ***N*M*** pieces. She can choose the order in which she gives Peter any remaining blocks, and she can also tell Peter what cuts to make (horizontal or vertical) and where exactly to make them. Help Bonny decide how to cut the chocolate into individual pieces, so that she pays Sly Peter as few raisins as possible.

TASK
Write a program that, given the number of raisins on each of the individual pieces, determines the minimum number of raisins that Bonny will have to pay Sly Peter.

CONSTRAINTS
1 <= ***N, M*** <= 50 The number of pieces on each side of the chocolate
1 <= ***Rk,p*** <= 1000 The number of raisins on the piece in the ***kth*** row and the ***pth*** column

INPUT
Your program must read from standard input the following data:
· The first line contains the integers ***N*** and ***M***, separated by a single space.
· The next ***N*** lines describe how many raisins there are on each piece of the chocolate. The ***kth*** of these ***N*** lines describes the ***kth*** row of the chocolate. Each such line contains ***M*** integers separated by single spaces. The integers describe the pieces on the corresponding row in order from left to right. The ***pth*** integer on the ***kth*** line (among these ***N*** lines) tells you how many raisins are on the piece in the ***kth*** row and the ***pth*** column.

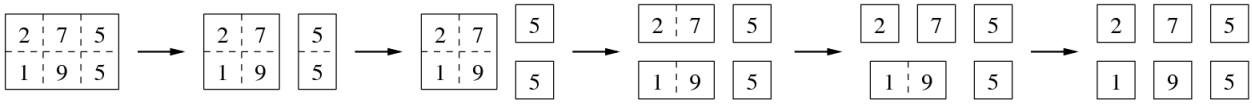
OUTPUT
Your program must write to standard output a single line containing a single integer: the minimum possible number of raisins that Bonny would have to pay Sly Peter.

EXAMPLE

Sample Input
2 3
2 7 5
1 9 5

Sample Output
77

One possible way (out of many) to achieve a cost of 77 is as follows:



The first cut that Bonny asks Peter to make separates the third column from the rest of the chocolate. Bonny needs to pay Peter 29 raisins for this.

Then Bonny gives Peter the smaller of the two blocks: the one that has two pieces with 5 raisins each, and asks Peter to cut the block in two in exchange for 10 raisins.

After this, Bonny gives Peter the largest remaining block: the one having pieces with 2, 7, 1 and 9 raisins respectively. Bonny asks Peter to cut it horizontally, separating the first and the second row and pays him 19 raisins.

Following this, Bonny gives Peter the top-left block, paying 9 raisins. Finally, Bonny asks Peter to split the bottom-left block, paying 10 raisins.

The total cost to Bonny is 29 + 10 + 19 + 9 + 10 = 77 raisins. No other cutting arrangement can get the chocolate cut into its 6 pieces at a smaller cost.

Towers

Little Johnny has built a number of towers of various heights using building bricks. Each brick has an integer number written on it. Johnny would like to construct a tower with the greatest possible sum of numbers written on bricks this tower is composed of; however, he does not want to completely destroy towers already constructed. Therefore, he decided that the only thing he will do is performing the following operation on pairs of towers of different heights:

- remove from the higher tower the l top blocks, where l is the height of the shorter tower, and create from them a new tower of height l (without changing the order of the blocks);
- afterwards, take the shorter tower and put it on top of the one we removed blocks from.

In this way, the two newly constructed towers have the same heights as the towers before the operation, but numbers written on corresponding blocks may be different.

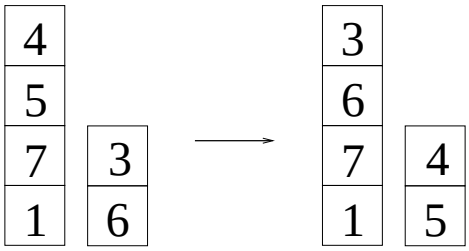


Figure: One operation on towers (4, 5, 7, 1) and (3, 6).

Johnny would like to maximize the value of the most valuable tower, where the value of a tower is the sum of numbers written on bricks it is composed of. He can make as many operations as he wants to.

Input

The first line of the input contains a single integer n ($1 \leqslant n \leqslant 500\,000$) — the number of towers built by Johnny. Each of the next n lines contains a description of one tower. The $(i+1)$-th line contains an integer w_i ($1 \leqslant w_i \leqslant 1\,000\,000$) — the height of the i-th tower — followed by w_i space-delimited integers $x_1, x_2, \ldots, x_{w_i}$ ($-1\,000\,000 \leqslant x_k \leqslant 1\,000\,000$). The number x_k is the number written on the k-th block (counting from the top) of the i-th tower. You may assume that the total number of blocks in all towers does not exceed 1 000 000.

Output

Your program should output a single line containing a single integer — the maximum possible value of a tower that Johnny can build.

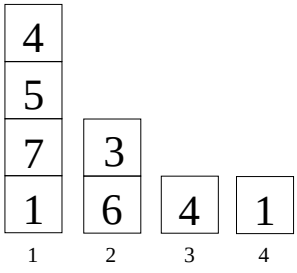
Example

For the input data:

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

the correct result is:

```
18
```



Explanation of the example: Johnny can use towers 2 and 3 to build the tower (4, 6). Afterwards, he can use this tower together with tower number 1 to build the tower (4, 6, 7, 1), which has value equal to 18.

Keyboard

Byteman has received an extraordinary keyboard as a gift. There are $n \cdot m$ keys on it, placed in n rows with m columns each. Moreover, all keys except the one in the top left corner are covered with domino tiles of dimensions 1×2, so that there are $\frac{n \cdot m - 1}{2}$ domino tiles in total. At any time, Byteman can move onto the free key one of the domino tiles adjacent to it by the shorter side. He can also press keys, but only if they are not covered.

Byteman would like to test (i.e., press) all keys with vowels, that is, letters `a`, `e`, `i`, `o`, `u` or `y`. What is the minimum number of tile moves neccessary to do that?

Input

The first line of the input contains two integers n and m ($1 \leqslant n, m < 70$) — the dimensions of the keyboard. The next n lines contain m lowercase letters of the English alphabet each, describing the rows of the keyboard. Each of the next n lines contains m characters describing placement of the domino tiles: `.` (ASCII code 46) denotes an uncovered key, `-` (ASCII code 45) denotes a key covered by a domino tile placed horizontally and `|` (ASCII code 124) — a key covered by a tile placed vertically.

Output

If it's not possible for Byteman to press all keys with vowels, your program should output just the single word "`NIE`". Otherwise output the minimum number of tile moves that Byteman must make in order to press all keys with vowels.

Example

For the input data:

```
3 3
ytr
hgf
dsa
.--
|||
|||
```

the correct result is:

```
2
```