2010 National Collegiate Programming Contest

- Problems: There are 7 problems (14 pages in all, not counting this cover page) in this packet.
- Program Input: Input to the programs are through the input files. Input filenames are given in the table below. Each input file may contain one or more test cases. Test cases may be separated by any delimiter as specified in the problem statements.
- Program Output: All output should be directed to standard output (screen output).
- Time Limit: The judges will run each submitted program with certain time limit (given in the table below).

Table 1: Problem Information Sheet

	Problem Name	Input File	Time Limit
Problem A	Intersection	pa.in	5 sec.
Problem B	Dominant Matrix	pb.in	1 sec.
Problem C	Survival Ratio	pc.in	1 sec.
Problem D	Center of a City	pd.in	5 sec.
Problem E	Jobs	pe.in	10 sec.
Problem F	Where to Go for Vacation	pf.in	1 sec.
Problem G	Non-Separating Path	pg.in	3 sec.

Problem A
Intersection
Input File: *pa.in*
Time Limit: *5 seconds*

To recover 3D depth information from a stereo pair of image is important to mobile robot navigation. Before using triangulation to calculate the depth, one has to make sure that two lines are intersecting at one point. The stereo image to recover the 3D depth information is illustrated in Figure 1.

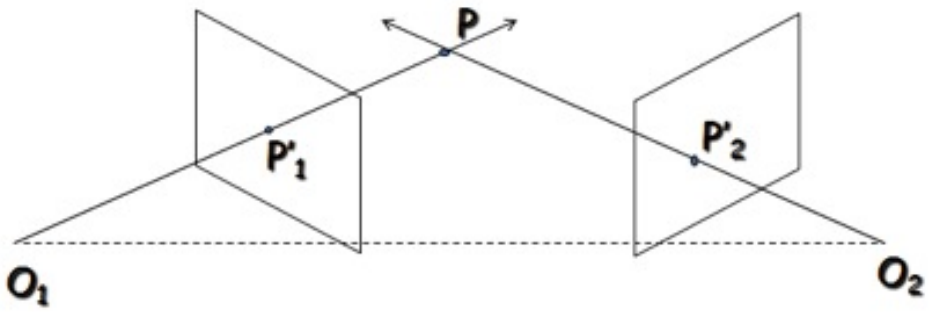


Figure 1: A stereo pair of images.

Given two or more lines, the goal is to find whether these two or more lines are all intersecting at one point.

Input File Format

The input consists of a number of test cases. The first line contains two positive numbers n, m to indicate the number of test cases to follow and the number of lines for each test case, respectively. For each test case, there are m lines of input. Each line contains six real numbers, $(x_1, y_1, z_1)$and (x_2, y_2, z_2), denoting two points on the given 3D line. The accuracy for the intersection point is up to 0.001.

Output Format

The output should indicate the order of case at the first line of each case, followed by Yes or No to indicate whether the given lines are intersected at one point. If Yes, also output the intersection point (X, Y, Z) with the accuracy rounded to 0.1.

Sample Input

```
3 4
1.0 2.0 3.0 0.0 0.0 0.0
1.0 2.0 3.0 5.0 0.0 0.0
1.0 2.0 3.0 3.0 4.0 0.0
1.0 2.0 3.0 4.0 3.0 2.0
4.0 3.0 3.0 1.0 2.0 3.0
4.0 3.0 3.0 3.0 4.0 5.0
```

```
8.0 6.0 6.0 6.0 8.0 10.0
2.0 3.0 1.0 4.0 5.0 9.0
2.0 2.0 2.0 3.0 4.0 6.0
2.0 2.0 2.0 30.0 20.0 30.0
3.0 2.0 2.0 11.0 2.0 3.0
2.0 5.0 2.0 12.0 20.0 30.0
```

Output for the Sample Input

```
case 1
Yes 1.0 2.0 3.0
case 2
No
case 3
No
```

Problem B
Dominant Matrix
Input File: *pb.in*
Time Limit: *1 second*

A real square matrix $A \in R^{n \times n}$, where $A = [a_{ij}],\ 1 \leq i, j, \leq n$, is said to be a *diagonally dominant matrix* if

$$|a_{ii}| \ > \ \sum_{j=1, j \neq i}^{n} |a_{ij}|, \quad 1 \leq i \leq n$$

This problem asks you to write a program to detect if a real square matrix is diagonally dominant.

Input File Format

The first line of the input file always contains one integer K indicating the number of test cases to come. Each test data set consists of $n + 1$ lines, where the first line gives the dimension of the matrix, followed by n lines, each line contains n floating-point numbers with the format "6.1f", which is a matrix to be detected. Note that $K \leq 10$ and $n \leq 8$ in this problem.

Output Format

A single line of $K \leq 10$ consecutive characters consisted of Y and N will be reported, where Y means a matrix is diagonally dominant and N means the tested matrix is not diagonally dominant.

Sample Input

```
4
3
  -4.1   1.2   2.8
   1.9   3.1  -1.1
   3.0  -2.7   5.8
4
   5.0   1.0  -2.0  -1.9
  -2.9   7.0  -2.0  -2.0
   1.1   1.2   4.0   1.5
   2.2   3.3  -4.4   5.5
5
   5.0   4.0   0.9   0.0   0.0
   0.0   7.0   0.9   2.1   3.9
   1.0   4.0   9.1   3.0   1.0
   0.3   1.0   9.2  19.9  -9.3
   3.1   4.2   5.3   2.4  15.1
8
   8.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0
  -1.0   8.0   1.0   1.0   1.0   1.0   1.0   1.0
```

```
 -1.0  -1.0   8.0   1.0   1.0   1.0   1.0   1.0
 -1.0  -1.0  -1.0   8.0   1.0   1.0   1.0   1.0
 -1.0  -1.0  -1.0  -1.0   8.0   1.0   1.0   1.0
 -1.0  -1.0  -1.0  -1.0  -1.0   8.0   1.0   1.0
 -1.0  -1.0  -1.0  -1.0  -1.0  -1.0   8.0   1.0
 -1.0  -1.0  -1.0  -1.0  -1.0  -1.0  -1.0   8.0
```

Output for the Sample Input

```
YNYY
```

Problem C
Survival Ratio
Input File: *pc.in*
Time Limit: *1 second*

Cartesian is a moon of the gas giant Polyphemus, which orbits Alpha Centauri A. Discovered by the first inter-stellar expedition in 2129, Cartesian has been mostly neglected due to the lack of diversified life forms like its more famous sibling moon named Pandora. Despite the harsh condition on Cartesian, bountiful water and living creatures are discovered on the planet surface. One of the creatures is a sort of intelligent one dimensional life forms live on non-overlapping triangular formations with one edge coincident with the normal water level. As shown in Figure 1, we can see five of such habitations for these creatures. All creatures live on the edges of the triangles, and they can not survive under water; hence, each of these triangles is a closed community on its own. To make matters worse, they are facing extreme climate changes, and after thorough studies, it is determined that if greater than or equal to 90 percent of the triangle is submerged under water then the habitation will cease to exist, i.e., all creatures on that triangle will eventually die out. You are to write a program, given the layout of the planet Cartesian and a sequence of water levels, calculate and print out the number of surviving habitations.

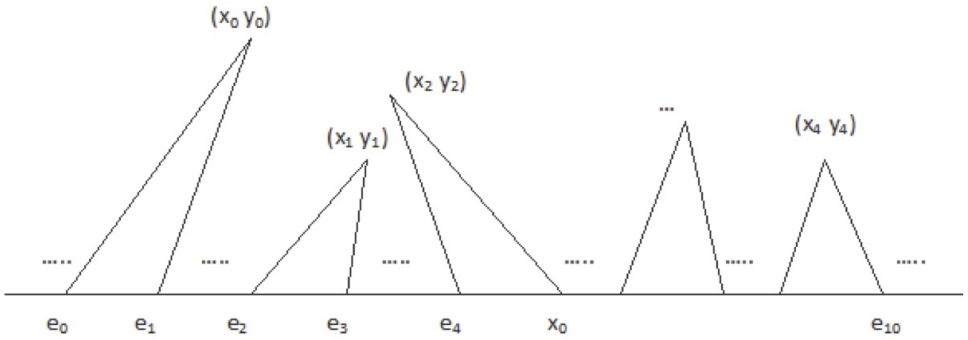


Figure 1: Sample habitations on planet Cartesian, with the horizontal line presents the normal water level.

Given two or more lines, the goal is to find whether these two or more lines are all intersecting at one point.

Input File Format

The input file is a text file, it contains a sequence of data lines with 4 lines of text describing a single test case. There is no limit on the number of lines that any input file can have, the program should keep on processing test cases until the end-of-file is reached, should there be inconsistency in the data describing any test case, a '0' should be reutned as the result. For each test case, the first line is a number, n ($1 \leq n \leq 20$), indicating the number of habitations. The second line, contains $2n$ integers in ascending order, $e_0, e_1, ..., e_{2n-2}, e_{2n-1}$, defining the base regions of the habitations at the normal water level, i.e., $(e_0, 0)$ – $(e_1, 0)$ is the base of the first habitation and $(e_{2n-2}, 0)$ – $(e_{2n-1}, 0)$ is the base of the n^{th} habitation. The third line, contains $2n$ integers, $x_0, y_0, x_1, y_1, ..., x_{n-1}, y_{n-1}$, which are the peak points of each habitation. The fourth line is a sequence of k ($1 \leq k \leq 20$) integers, $h_0, h_1, ..., h_{k-1}$,

which are the proposed water levels.

Output Format

For each test case, output a sequence of k integers, $s_0, s_1, ..., s_{k-1}$, where s_i is the number of survived habitations with water level h_i.

Sample Input

```
3
1 2 4 5 7 9
1 4 3 8 10 6
3 6 9
2
2 4 8 10
1 10 5 20
9 16 18
```

Output for the Sample Input

```
3 1 0
1 1 0
```

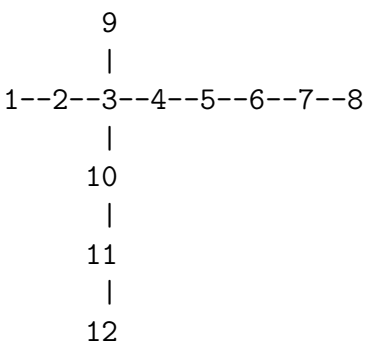
Problem D
Center of a City
Input File: *pd.in*
Time Limit: *5 seconds*

Herbert is a billionaire who wants to hire the best programmers in the world and open a software service company near the center of a big city. Although the city is big, it has a very convenient subway system. Herbert wants to buy an office near a subway station which can reach every part of the city quickly by subway.

For simplicity, assume that one can go from any subway station to any other subway station, and the subway system contains no cycles.

The center of the city where Herbert wants to buy an office nearby is define as the subway station which can go to any other station with minimum number of stops.

For example, consider a city where the subway system is shown in the following figure.



The center of this city is defined as the subway station 4 since it can reach any other station by at most 4 stops. All other stations need at least 5 stops to reach the farthest station.

You are going to write an efficient program to solve the problem with a large number of subway stations.

Input File Format

An instance of the problem consists of

1. the number of subway stations n, and
2. the description of the subway system.

Since the subway system is connected and contains no cycles, it can be described by a rooted tree.

1. Select any station as the root.
2. For each non-root station, specify its parent station.

Assume that the stations are numbered by $1, 2, \ldots, n$. Without loss of generality, assume that station 1 is the root. Each instance is described by n integers, and they are stored in $\lceil (n-1)/20 \rceil + 1$ lines in the input file.

1. The first line is the integer n.
2. The following lines contains $n-1$ integers, at most 20 integers in a line.

Let the number n be the first integer. Suppose that the i-th integer is j. This means that the parent station of station i is station j. In this problem, we assume that $1 \leq n \leq 10000$. If $n = 1$, then station 1 is the center.

Notice that the test data file may contain more than one instances. The last instance is followed by a line containing a single 0.

Output Format

The outputs for each test case is the center of the city, which has been defined above. If there are more than one centers print all the centers.

Sample Input

```
12
1 2 3 4 5 6 7 3 3 10 11
6
1 2 3 4 5
0
```

Output for the Sample Input

```
4
3 4
```

Problem E
Jobs
Input File: *pe.in*
Time Limit: *10 seconds*

We want to schedule n jobs to run on three types of machines. These job must run sequentially, i.e., we can do only one job at a time, and the i-th job cannot start until the $i-1$-th job finishes. We assume that each job requires one among many possible machine combinations of machine A, B, and C. For example, a job j_i may require (a) two machine A, or (b) one machine A, one machine B, and one machines C. To simplify the notation we use $(2,0,0)$ for the first combination and $(1,1,1)$ for the second combination. Let the i-th job be j_i so job j_i has a set of vectors of these possible machine configurations. For example, the set of possible machine configurations for job j_i could be $\{(2,0,0),(1,1,1)\}$. We also assume that even when a job j_i can run in several machine configurations, the execution time i_i remains the same.

We want to minimize the total cost to run all jobs. We assume that it costs $\mathcal{A}$ amount of money to use a machine A for an unit of time. Similarly we define $\mathcal{B}$ and $\mathcal{C}$ accordingly. In addition, once you start using a machine, you must pay for the machine until all jobs finish, even for the time that you are not using the machine.

Now given the number of jobs (n), possible machine configurations for all jobs (m_i), the execution time t_i for all jobs, and the cost of using a machine A, B, or C for an unit of time, please compute the *minimum* cost to finish all jobs. We assume that the number of all three machines are unlimited.

Technical Specification

1. There are n jobs, where $1 \leq n \leq 30$.
2. Each job j_i has m_i possible machine configurations, where $1 \leq m_i \leq 10$.
3. The execution time of job t_i is a positive integer no more than 100.
4. The costs of using a machine for an unit of time are $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{C}$, where $1 \leq \mathcal{A},\mathcal{B},\mathcal{C} \leq 100$.
5. Let r_a, r_b, and r_c be the number of machines in a machine configuration, then $0 \leq r_a, r_b, r_c \leq 200$.

Input File Format

The input file contains a set of test data and the first line of the input is the number of test data. Each test data consists of two parts. The first part has one line containing the number of jobs n, and $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$ – the cost to use a machine A, B, or C for an unit of time respectively. The second part has n lines. The i-th line starts with t_i and m_i, then the m_i machine configurations that could run job j_i.

Output Format

The output for each test case is a single line containing the minimum cost to run all jobs for that test case.

Sample Input

```
1
2 3 4 5
1 2 0 0 1 0 1 0
2 1 0 0 1
```

Output for the Sample Input

```
15
```

Problem F
Where to Go for Vacation
Input File: *pf.in*
Time Limit: *1 second*

Recent news reported that about 60% of people working in the high-tech companies want to leave their jobs. These people are tired of long working time and high pressure. The CEO of Happy Software read this news and decided to offer vacation for his employees. The coming question is where to go for vacation. The CEO selects many beautiful cities like Paris, Los Angels, London, and so on. He thinks that employees in the same division should have vacation to the same city so that they can have common experiences and topics to talk about. The employees in different divisions should go to different cities so that they can share souvenirs and photos. Besides, in each division only one employee can leave for vacation at a time; otherwise, the performance of the division will be too low.

For an employee i, he/she has a desired staying duration t_{ij} for each city j. Since there is a big project to be completed soon, each employee i must finish his/her vacation and come back within d_i days from now on. The value of d_i depends on the importance of the employee to the coming project. A division cannot go to a city if there exists no feasible schedule for every member to come back in time. If a division has no place to go for vacation (due to place collision with other divisions or vacation deadlines of members), the company will give \$50,000 compensation to the division. Now the CEO asks for your help to write a program to determine the minimum amount of compensation he needs to pay.

Technical Specification

1. There are M cities to be arranged, where $1 \leq M \leq 100$.
2. There are N divisions, where $1 \leq N \leq 100$.
3. Each division k consists of n_k employees, where $1 \leq n_k \leq 10$. (Each employee belongs to exactly one division.)
4. For the staying duration t_{ij}, $1 \leq t_{ij} \leq 10$.
5. For the deadline d_i, $1 \leq d_i \leq 100$.

Input File Format

The first line of the input file contains an integer, denoting the number of test cases to follow. For each test case, the first line contains two positive integers M and N. The next line contains the number of employees n_1 in division 1, followed by n_1 lines for employee i ($i = 1, 2, ..., n_1$), each containing $(M + 1)$ positive numbers, where the first one is the deadline d_i and the remaining are t_{ij} ($j = 1, 2, ..., M$). Similarly, the next $(n_2 + 1)$ line contains the number of employees n_2 in division 2 and data of employees in this division. Data of the remaining $(N - 2)$ divisions are also given in the same format. The next test case is separated from the previous one by one empty line.

Output Format

For each test case, output the minimum amount of compensation on one line.

Sample Input

```
2    // two test cases
2 2  // two cities and two divisions
3    // three employees in division 1
10 2 2
6 3 3
4 2 3
2    // two employees in division 2
5 4 3
7 3 4

3 3
2
5 2 4 2
3 4 2 3
1
6 3 3 3
3
12 5 5 5
8 3 6 4
6 5 3 2
```

Output for the Sample Input

```
0
50000
```

Problem G
Non-Separating Path
Time Limit: *3 seconds*

Let $G = (V, E, w)$ be an undirected connected graph with nonnegative node weight w. For two vertices s and t, an st-path is a path $P = (v_0 = s, v_1, v_2, \ldots, v_l = t)$ from s to t. In this problem, a path must be a simple path, i.e., $v_i \neq v_j$ for any $0 \leq i < j \leq l$. A path is "non-separating" if the remaining graph is still connected after removing the path. Precisely speaking, an st-path P is non-separating if the subgraph induced by $V - V(P)$ is connected, in which $V(P)$ denotes the node set of P. The weight of a path is the total weight of the nodes on the path. It is known that finding a minimum weight non-separating path in general graphs is an NP-hard problem but it can be efficiently solved for "grid graphs". In this problem, you are asked to design a program for finding minimum weight non-separating paths in grid graphs.

An $m \times n$ grid graph M can be thought of as an $m \times n$ matrix such that $V(M) = \{M_{ij} | 0 \leq i < m, 0 \leq j < n\}$ and there exists an edge between two consecutive nodes in the same row or the same column. The set of nodes in the first or the last row and the first or the last column is called as the *boundary* of the grid graph. As you can see, each node except for the boundary nodes has exactly four neighbors. We assume that s must be on the boundary and $m, n \geq 3$.

For example, the left and the right parts of Figure 1 are the same 4×3 grid graph, in which each node is labeled by its weight. In the left, the bold line is a minimum weight st-path but it is not a non-separating path. In the right, the bold line shows the minimum weight non-separating st-path. For this instance, the minimum weight of any non-separating st-path is 15.

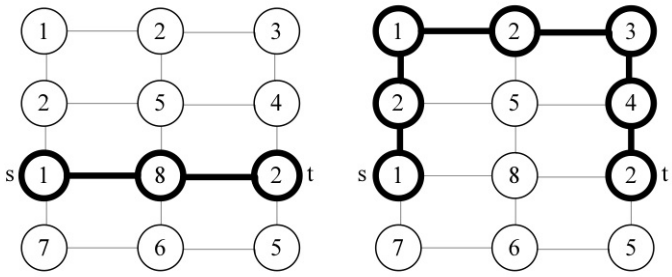


Figure 1: A separating path and a non-separating path in a grid graph

Technical Specification

1. The number of nodes in each grid graph is bounded by 40000, i.e., $mn \leq 40000$.
2. The weight of any node is a nonnegative integer and at most 100000.
3. The total weight of all nodes in a grid graph is at most 2^{31}.
4. $m \geq 3$ and $n \geq 3$.

Input File Format

There are several test cases. For each test case, the first line contains two integers m and n which are the numbers of rows and columns, respectively. The next m lines contains the weights of the nodes in an $m \times n$ grid graph M as follows:

$$
\begin{array}{lllll}
w(M_{0,0}) & w(M_{0,1}) & w(M_{0,2}) & \ldots & w(M_{0,n-1}) \\
w(M_{1,0}) & w(M_{1,1}) & w(M_{1,2}) & \ldots & w(M_{1,n-1}) \\
\ldots & \ldots & & & \\
w(M_{m-1,0}) & w(M_{m-1,1}) & w(M_{m-1,2}) & \ldots & w(M_{m-1,n-1})
\end{array}
$$

The last two lines of each test case are the row and column indexes of the nodes s and t, one line for one node. The row index is an integer between 0 and $m-1$. The column index is an integer between 0 and $n-1$. The input ends with a case that $m = 0$ and $n = 0$.

Output Format

For each test case, output the minimum weight of any non-separating path from s to t in one line.

Sample Input

```
4 5
5 6 3 2 4
4 1 8 1 5
3 1 1 1 5
2 2 2 2 2
0 3
1 1
4 3
1 2 3
2 5 4
1 8 2
7 6 5
2 0
2 2
3 4
20 1 1 1
1 1 1 1
1 1 1 1
0 1
1 0
0 0
```

Output for the Sample Input

```
7
15
11
```