2014 Taiwan National Collegiate Programming Contest

October 18. 2014

- Problems: There are 12 problems (27 pages in all) in this packet.
- Problem Input: Input to the problems are through standard input. Each input file may contain one or more test cases. Test cases may be separated by any delimiter as specified in the problem statements.
- Problem Output: All output should be directed to the standard output (screen output).
- Time Limit: Judges will run each submitted program with certain time limit (given in the table below).

Table 1: Problem Information

	Problem Name	Time Limit
Problem A	Maximum Number of Drinks	1 sec.
Problem B	Sequence Difference Check	1 sec.
Problem C	Solve Quadratic Equations	1 sec.
Problem D	Line Detecting	1 sec.
Problem E	Desert	5 sec.
Problem F	A Tunnel Game	1 sec.
Problem G	Bigraphical Sequence of Numbers	1 sec.
Problem H	Classrooms	3 sec.
Problem I	Determinant of a Square Matrix	3 sec.
Problem J	Chromatic Polynomials	1 sec.
Problem K	Surveillance and Risk	3 sec.
Problem L	The Maximum WN-Shape Sum	3 sec.

Problem A
Maximum Number of Drinks
Time Limit: *1 Second*

A soft drink company is now promoting the sales of a special type of drinks. If you collect 4 empty bottles or 3 caps of this type of soft drinks you can get 1 for free. For example, suppose you buy 3 bottles of the soft drinks. After drinking all of them, you will have 3 empty bottles and 3 caps. You can then exchange the 3 caps for 1 free soft drinks. After drinking this free soft drinks, you will have 4 empty bottles and 1 cap. Using the 4 empty bottles, you can get another free soft drink. Thus, you can have 5 soft drinks, and still have 1 empty bottle and 2 caps. It seems easy, right? Not quite. Can you figure out, by mental arithmetic, how many soft drinks you can have if you buy 20 soft drinks?

Write a program to solve this type of problems. Namely, initially you buy a bottles of soft drinks. After drinking all of them, you can exchange b empty bottles for a free soft drink. You can also exchange c caps for a free soft drink. Do it repeatedly. What is the maximum number of soft drinks you can have? How many empty bottles and caps are left?

Input Format

The input to the problem is a file containing many test cases. Each test case is in one line, which contains three integers a, b, and c; meaning that initially you buy a soft drinks, and b empty bottles or c caps can get 1 free soft drink. Assume that $0 < a < 2^{31}$, $b, c > 1$. The last line of the input file contains a single 0. You do not need to process this line.

Output Format

The outputs for each test case contains 3 integers: (1) maximum number of soft drinks you can have, (2) number of empty bottles left, and (3) number of caps left. Print these 3 numbers in one line, with a space between adjacent integers.

Sample Input

```
3 4 3
20 7 4
0
```

Sample Output for the Sample Input

```
5 1 2
31 3 3
```

Problem B
Sequence Difference Check
Time Limit: *1 Second*

Consider a sequence of N integers: $\langle a_1, \ldots, a_N \rangle$. We call this sequence *nice* if $|a_{i+1} - a_i| \in \{1, 2, \ldots, 2^{10}\}$, for $i = 1, \ldots, n-1$. In other words, the sequence is nice, if the absolute values of the difference between successive numbers are power of 2, up to 2^{10}.

For example: $\langle 1, 5, 21, 17 \rangle$ is a nice sequence of length 4. While the sequence $\langle 1, 6, 22 \rangle$ is not a nice one, because the difference between 1 and 6 is 5, which is not a power of 2. Write a program to determine whether a sequence is a nice one or not.

Technical Specification

1. All the numbers are non-negative integers.
2. K: the number of test cases. $K \leq 20$.
3. N: the size of sequence. $N \leq 2000$.

Input File Format

The first line of the input file contains an integer $K(\leq 20)$ indicating the number of test cases to follow. Each test case starts with a positive integer: N indicating the number of integers. Then N integers follow and adjacent integers are separated with space(s).

Output Format

For each test case, output YES, if the sequence is nice; otherwise output NO.

Sample Input

```
2
4 1 5 21 17
3 6 1 33
```

Output for the Sample Input

```
YES
NO
```

Problem C
Solve Quadratic Equations
Time Limit: *1 Second*

It is well-known that the solution to the quadratic equation $ax^2 + bx + c = 0$, is

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

In this problem, you are going to write a program to solve quadratic equations *symbolically*. That is, print exact solutions, not floating point approximations. For example, the solution to the equation $x^2 + 2 = 0$, should be $x = \pm\sqrt{2}\texttt{i}$, not $x = \pm 1.41421356237\ldots\texttt{i}$.

Printing the output of the solutions to a text terminal can be hard. For example, square root, fractions, etc, are not easy to print in a text terminal. Therefore, we will simplify the output format so that each solution can be printed in one line. In general, print the solutions by using the format

$$x1 = (\alpha + \beta\texttt{\textbackslash sqrt\{}\gamma\texttt{\}i})/\delta$$
$$x2 = (\alpha - \beta\texttt{\textbackslash sqrt\{}\gamma\texttt{\}i})/\delta$$

If the solutions are real numbers, there will be no `i`'s.

In this problem, assume that all the coefficients a, b, and c are integers, and $a \neq 0$. Note that

1. α, β, γ, and δ must be integers,
2. $\gcd(\alpha, \beta, \delta) = 1$,
3. β and γ must be positive, and
4. γ must be square-free.

An integer is *square-free* if it is divisible by no perfect squares, except 1. For example, 6 is square-free, but $12 = 2^2 3$ is not. Therefore, $\sqrt{12}$ should be converted to $2\sqrt{3}$.

Furthermore, simplify the output by using the following rules. Do it until no more simplifications can be done.

1. $0 + x = x + 0 = x$.
2. $1 \times x = x \times 1 = x$.
3. $x/1 = x$, $x/(-y) = -x/y$.
4. Any fraction x/y must be simplified, i. e., $\gcd(x, y) = 1$.

The output should also satisfies the following requirements.

1. No multiplication symbols ($\times$, `*`, etc.) are printed.
2. Add necessary "(" and ")" to make the solution correct, but no extra "(" and ")" can be added.
3. Print real part first for complex numbers, e. g. $a + b\texttt{i}$ or $a - b\texttt{i}$.
4. No spaces are required, unless it must be used to separate two terms.
5. If the equation has only one solution, print only one solution $x = \ldots$.
6. If the equation has no solutions, print "`no solutions`".

Input Format

The input to the problem is a file containing many test cases. Each test case is written in one line, which contains three integers a, b, and c. Assume that $|a|, |b|, |c| < 2^{20}$, and $a \neq 0$. The last line of the input file contains a single 0. You do not need to process this line.

Output Format

Print the solutions to the quadratic equation $ax^2 + bx + c = 0$ symbolically by using the format as described above. Print each solution in one line. Print a blank line after the last solution of each test case.

Sample Input

```
1 2 1
1 -3 2
1 3 1
1 1 1
1 0 1
0
```

Sample Output for the Sample Input

```
x=-1

x1=1
x2=2

x1=(-3+\sqrt{5})/2
x2=(-3-\sqrt{5})/2

x1=(-1+\sqrt{3}i)/2
x2=(-1-\sqrt{3}i)/2

x1=i
x2=-i
```

Problem D
Line Detecting
Time Limit: *1 Second*

Consider N lines of strings, where each string has length N and the string consists of only two kinds of characters: '.' and '+'. You can think of these N lines of strings as a picture. Your task is to write a program to determine whether the picture has a straight line formed by the character '+'. If there is no such line, then output 'Not a line'; otherwise output its slope, which can be calculated as follows. For any two '+' characters, one is in the i-th row and j-th column and the other is in the i'-th row and j'-th column. Then the slope is $\frac{j'-j}{i'-i}$, which is ∞, when $i = i'$. If the '+' characters form a line, then the slope is a constant for any two '+' characters on that line.

Technical Specification

1. There are at least two '+' characters in each picture.
2. The slope will be an integer or ∞, if the '+' characters form a line.
3. K: the number of test cases. $K \leq 20$.
4. N: the number of strings and the length of each string. $N \leq 500$.

Input File Format
The first line of the input file contains an integer $K(\leq 20)$ indicating the number of test cases to follow. Each test case starts with a positive integer: N indicating the number of strings and the string length. Then N strings follow and each has N characters.

Output Format
For each test case, output the slope, if the picture has a line of '+'; otherwise output 'Not a line'. If the slope is ∞, then output 'INF'.

Sample Input

```
3
5
.....
+++++
.....
.....
.....
10
.........+
.......+..
.....+....
...+......
.+........
..........
..........
..........
..........
..........
10
+.........
+......+..
+.........
+...+.....
+.........
+.........
+....+....
+.........
+.........
+.........
```

Output for the Sample Input

```
INF
-2
Not a line
```

Problem E
Desert
Time Limit: *5 Seconds*

A gold seeker wants to travel through a desert and collect gold. The desert is an m by n grid, and the gold seeker will start his journey from the top left corner $(0,0)$ and finish it at the bottom right corner $(m-1, n-1)$. During each day of his journey, if he starts from grid (i,j) then he can either go south to grid $(i,j+1)$ if $j < n-1$, or go east to grid $(i+1,j)$ if $i < m-1$. Each such move will consume one unit of fuel on his jeep. The desert is huge so the gold seeker has already reserved fuel in some grids. When he reaches a grid he can refuel his jeep with all the fuel in that grid, but note that his jeep can carry only C units of fuel. In addition, some grids have gold where the gold seeker can only collect if he reaches those grids. Your task is to help the gold seeker plan his route so that he can finish his journey and collect the maximum amount of gold.

Technical Specification

1. $2 \leq m, n \leq 1,000$, where m and n are the width and length of the desert.
2. $1 \leq C \leq 100$, where C is the maximum units of fuel the gold seeker's jeep can carry.
3. $0 \leq f_{i,j} \leq c$, where $f_{i,j}$ is the units of fuel in grid (i,j).
4. $0 \leq g_{i,j} \leq 100$, where $g_{i,j}$ is the amount of gold in grid (i,j).

Input File Format
The first line of the input file has the number of test cases. Each test case is as follows. The first line of a test case contains three integers m, n, and C. Each of the following n lines has m integers as $f_{i,j}$. Each of the next following n lines has m integers as $g_{i,j}$.

Output Format
The output of each test case has an integer, which indicates the maximum amount of gold that the gold seeker can collect by traveling from $(0,0)$ to $(m-1, n-1)$. If there is no way for the gold seeker to reach the destination, then output -1.

Sample Input

```
1
4 3 20
2 0 0 0
0 3 0 0
100 0 0 1
0 0 80 80
0 1 10 10
0 0 0 5
```

Output for the Sample Input

```
26
```

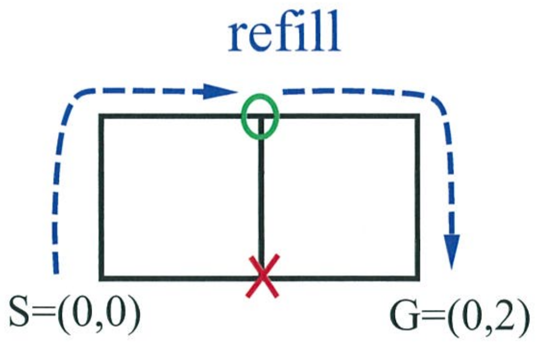
Problem F
A Tunnel Game
Time Limit: *1 Second*

Deep under the ground, an ancient civilization built a tunnel system and hide a massive amount of treasure in one place. After a careful survey using advance radar systems, it is found the system can be modeled as a 2-dimensional n by m grid. The distance between a grid point to another adjacent grid point is the same among all grid points. For ease of description, the grid points are numbered as (x, y), $0 \leq x < n$ and $0 \leq y < m$ with $(0, 0)$, $(n-1, m-1)$, $(0, n-1)$, and $(0, m-1)$ being the south-west, north-east, north-west, and south-east corners, respectively. The entry point $S = (x_s, y_s)$ of the system is known, and the place $G = (x_g, y_g)$ where the treasure is placed is also known.

The tunnels are filled with water and can only be entered by human divers. A treasure hunter that is a diver can only carry h units of oxygen in his/her scuba where each unit of oxygen can only be used to travel from one grid point to its adjacent grid point. The tunnels are well built without any faults after years, but some intersections, e.g., grid points, may be blocked because of the old age. It is also found that the ancient civilization left some secret oxygen generating devices at some grid points that a diver can use to refill his/her scuba. The device can be used many time and generate enough oxygen each time, but cannot be moved. Note that a treasure hunter can use the oxygen generating device the moment he/she arrives a grid point that has an oxygen generator to breath normally.

Your goal is to find the length of a shortest path from (x_s, y_s) to (x_g, y_g) so that when a treasure hunter with a full scuba initially can reach the destination safely with oxygen to breath along the way. Note that there is no water in G and there is a secret tunnel to the ground there. There is also a tunnel from the ground to the starting point S.

As an example, assume $n = 2$, $m = 3$, $h = 2$, $(x_s, y_s) = (0, 0)$, and $(x_g, y_g) = (0, 2)$. There is an oxygen generator at $(1, 1)$ and $(0, 1)$ is blocked. The treasure hunter must first go to $(1, 0)$, then $(1, 1)$ and refilled, then $(1, 2)$, and finally $(0, 2)$. The length of this path is 4.



Technical Specification

1. $0 < n \leq 1,000$
2. $0 < m \leq 1,000$
3. $0 < h \leq 100$

4. $0 \leq b \leq 100$
5. $0 \leq d \leq 100$

Input File Format The first line of the input file contains an integer indicating the number of test cases. For each test case, the first line are 9 integers indicating n, m, h, x_s, y_s, x_g, y_g, b and d where b is the number of grid points that are blocked, and d is the number of oxygen generators. The next line contains $2b$ integers indicating the location of the b grid points that are blocked with each pair of integers x and y representing the x and y coordinates of the blocked point. The next line contains $2d$ integers indicating the location of the b grid points that has an oxygen generators with each pair of integers x and y representing its X and Y coordinates.

Output Format For each test case, output the length of the shortest path from (x_s, y_s) to (x_g, y_g). If it is impossible to reach the goal, then output -1.

Sample Input

```
2
3 3 2 0 0 2 2 0 0


2 3 2 0 0 0 2 1 1
0 1
1 1
```

Output for the Sample Input

```
-1
4
```

Problem G
Bigraphical Sequence of Numbers
Time Limit: *1 Second*

A bipartite graph is a graph whose vertices can be divided into two disjoint sets U and V (that is, U and V are independent sets) such that every edge connects a vertex in U to one in V. The degree of a vertex v, denoted by $\deg(v)$, is equal to the number of vertices adjacent to v. For a bipartite graph G with vertex sets $U = \{u_1, \ldots, u_m\}$ and $V = \{v_1, \ldots, v_n\}$, the *bipartite degree sequence* contains two sequences $\deg(u_1)$, …, $\deg(u_m)$ and $\deg(v_1)$, …, $\deg(v_n)$ of nonnegative integers. The former is the degree sequence for the vertices in U and the latter for the vertices in V. We call two sequences of nonnegative integers *bigraphical* if it is the bipartite degree sequence of some bipartite graph. Note that there is at most one edge between any two vertices. You are asked to write a program to determine whether two sequences of nonnegative integers is bigraphical or not.

Technical Specification

- Each of two sequences of nonnegative integers contains at most 100 integers.
- Each integer in a sequence is less than or equal to 100.

Input File Format
The first line of the input file contains an integer denoting the number of test cases. There are at most 10 test cases. In each test case, the first line contains two integers, say m and n, denoting the numbers of integers in the two sequences. The second line contains a sequence of m nonnegative integers. The third line contains a sequence of n nonnegative integers. Note that there is a blank between any two integers in the sequence.

Output Format
For each test case, output 1 in a line if it is bigraphical; otherwise, output 0 in a line.

Sample Input

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

Output for the Sample Input

```
1
1
0
0
```

Problem H
Classrooms
Time Limit: *3 Second*

The CS department chair wants to schedule c classes (numbered 0 to $c - 1$) into two classrooms X and Y. Class i will use classroom X for x_i number of hours, and classroom Y for y_i number of hours, where both x_i and y_i are positive integers. The scheduling must follow four constraints.

- A class can use two classrooms in any arbitrary order, i.e., a class can use classroom X first then classroom Y, or classroom Y then classroom X.
- A class must use a classroom *without* interruption. Let bx_i and by_i be times class i starts using classroom X and Y respectively. Class i will use classroom X continuously from time bx_i to $bx_i + x_i$, and classroom Y continuously from time by_i to $by_i + y_i$.
- Only one class can use a classroom at a time. That is, for any class i there must not be another class j such that $[bx_j, bx_j + x_j)$ overlaps with $[bx_i, bx_i + x_i)$, or similarly $[by_j, by_j + y_j)$ overlaps with $[by_i, by_i + y_i)$.
- A class cannot use two classrooms simultaneously. That is, $[bx_i, bx_i+x_i)$ cannot overlap with $[by_i, by_i + y_i)$ for any class i.

Your task is to help the CS department chair determine the starting times for all classes on both classrooms (bx_i and by_i), so as to minimize the total time for all classes to complete. This schedule may not be unique and *any* such schedule is acceptable.

We use two examples to illiterate the constraints. We assume that classes will use classrooms as indicated in Table 2. The first schedule has $bx_0 = 0$, $by_0 = 1$, $bx_1 = 1$, $by_1 = 0$, and $bx_2 = 4$, $by_2 = 9$, and the total time is 11 hours (Figure ??). Note that class 2 uses classroom X from time 4 to 9, so it cannot use classroom Y until time 9. The second schedule has $bx_0 = 8$, $by_0 = 1$, $bx_1 = 5$, $by_1 = 0$, and $bx_2 = 0$, $by_2 = 5$, and the total time is 9 hours (Figure ??), which is the minimum total amount of time for all classes.

class	x_i	y_i
0	1	4
1	3	1
2	5	2

Table 2: An example of classes and the time for classrooms

Technical Specification

1. $1 \leq c \leq 10,000$, where c is the number of classes.
2. $1 \leq x_i, y_i \leq 100,000$, where x_i and y_i are the number of hours class i uses classroom X and Y respectively. It is also guaranteed that the total number of hours required is no more than 2147483647.

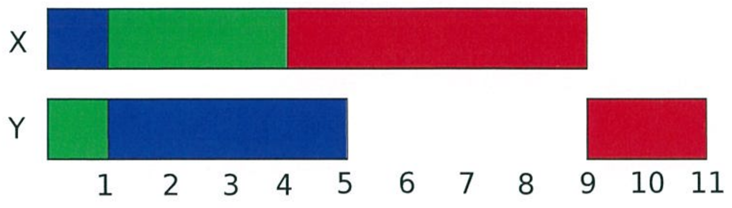


Figure 1: The first schedule

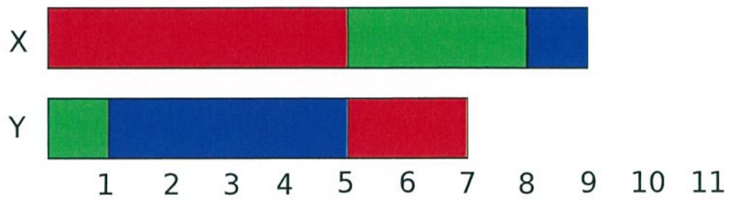


Figure 2: The second schedule

Input File Format

The first line of the input has the number of test cases. Each test case is as follows. The first line of test case contains a positive integer c. Each of the following c lines has two integers x_i and y_i.

Output Format

Each test case has the following output. The first line of a test case has a positive integer, which indicate the minimum number of hours required to schedule all classes. Each of the next c lines has two integers bx_i and by_i that indicate when class i will start in classrooms X and Y respectively.

Sample Input

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

Output for the Sample Input

```
9
8 1
5 0
0 5
```

Problem I
Determinant of a Square Matrix
Time Limit: *3 seconds*

The determinant of a matrix could be used in many applications including (1) to determine if a matrix is invertible, (2) to predict if a linear system has solutions, (3) to compute the inverse of a nonsingular matrix, and etc. The determinant of an $n \times n$ matrix A can be defined below.

Permutation: A *permutation* of integers $\{1, 2, \cdots, n\}$ is an ordered list of these n integers, denoted as $(a_1, a_2, \cdots, a_n)$, where $a_j \in \{1, 2, \cdots, n\}$, $1 \le j \le n$. A permutation is said to be *even* or *odd* according to whether the number of inversions of natural order of $1, 2, \cdots, n$ that are present in the permutation is even or odd, respectively. For example, when $n = 3$, (2,3,1) is an even permutation because 2 comes before 1 (but $1 \le 2$) and 3 comes before 1 (but $1 \le 3$) which counts two inversions; (3,2,1) is an odd permutation because there are 3 inversions involved which are 3 comes before 2, 3 comes before 1, and 3 comes before 2. All of the six permutations on $\{1, 2, 3\}$ are given below, the first line lists the 3 even permutations and the 2nd line lists the odd permutations.

$$(1,2,3) \quad (2,3,1) \quad (3,1,2)$$

$$(1,3,2) \quad (2,1,3) \quad (3,2,1)$$

Determinant: Let $A \in R^{n \times n}$ be denoted as $A = [a_{ij}]$, $1 \le i, j \le n$, the determinant of A is a function which can be regarded as a function $det : R^{n \times n} \rightarrow R$ defined as

$$det(A) = \sum_{\sigma} (-1)^{sign(\sigma)} \prod_{i=1}^{n} a_{i,\sigma(i)}$$

where σ is a permutation on $\{1, 2, \cdots, n\}$ and

$$sign(\sigma) = \begin{cases} 0 & if\ \sigma\ is\ even \\ 1 & if\ \sigma\ is\ odd \end{cases}$$

In particular, the determinant of a 2×2 matrix A and a 3×3 matrix B given below can be computed by using formulas provided as follows.

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}, \quad B = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix},$$

$$det(A) = a_{11}a_{22} - a_{21}a_{12}$$

$$\begin{aligned} det(B) &= b_{11}b_{22}b_{33} + b_{12}b_{23}b_{31} + b_{21}b_{32}b_{13} \\ &- b_{31}b_{22}b_{13} - b_{12}b_{21}b_{33} - b_{11}b_{32}b_{23} \end{aligned}$$

The computation of a large matrix determinant is generally based on the stragety of LU-decomposition with partial pivoting. Given an $n \times n$ matrix $A \in R^{n\times n}$, this problem asks you to write a program to compute the determinant of A, where $2 \leq n \leq 5$ and each element of matrix A is an integer in the range $[-32, 32)$.

Input File Format

The first line of the input file contains one integer, $K \leq 5$, indicating the number of test cases to come. The next $\sum_{j=1}^{K}(n_j + 1)$ lines consists of K sets of an integer of the dimension of the so-called matrix A followed by its entries of integers appearing row by row, integers in the same row are separated by space(s).

Output Format

The K integers corresponding to the determinants of the input matrices.

Sample Input

```
3
2
9 3
3 5
3
10 0 0
0 -20  0
0 0 30
5
-1 0 0 0 0
0 2 0 0 0
0 0 -3 0 0
0 0 0 4 2
0 0 0 2 -5
```

Output for the Sample Input

```
36
-6000
-144
```

Problem J
Chromatic Polynomials
Time Limit: *1 Second*

A company manufactures n chemicals. Some pairs of these chemicals are incompatible and may explode if brought into contact with each other. The company is going to build a container with x compartments to store these chemicals. Obviously incompatible chemicals should be assigned and stored in different compartments. To decide on the number of compartments the container should have, the company likes to know first the number of distinct valid assignments for placing n chemicals into x compartment container such that the incompatible chemicals will not be in the same compartment.

Note that each compartment should be regarded as distinct and can store many chemicals that are compatible. Two assignments are to be regards as different if at least one chemical is assigned to two different compartments in the two assignments.

Example (1): Given two chemicals C_1, C_2 ($n = 2$) which are compatible and a container with two compartments P_1 and P_2 ($x = 2$). There are four different valid assignments, namely

(1) $P_1 = \{\}$, $P_2 = \{C_1, C_2\}$,

(2) $P_1 = \{C_1, C_2\}$, $P_2 = \{\}$,

(3) $P_1 = \{C_1\}$, $P_2 = \{C_2\}$,

(4) $P_1 = \{C_2\}$, $P_2 = \{C_1\}$.

Example (2): Given two chemicals C_1, C_2 ($n = 2$) which are incompatible and a container with two compartments ($x = 2$) P_1 and P_2. There are only two different valid assignments, namely

(1) $P_1 = \{C_1\}$, $P_2 = \{C_2\}$,

(2) $P_1 = \{C_2\}$, $P_2 = \{C_1\}$.

Example (3): Given three chemicals C_1, C_2, C_3 ($n = 3$) and a container with two compartments ($x = 2$) P_1 and P_2. Suppose only C_1 and C_2 are incompatible, meaning that C_1 is compatible with C_3 and C_2 is also compatible with C_3. There are four different valid assignments, namely

(1) $P_1 = \{C_1\}$, $P_2 = \{C_2, C_3\}$,

(2) $P_1 = \{C_2\}$, $P_2 = \{C_1, C_3\}$,

(3) $P_1 = \{C_2, C_3\}$, $P_2 = \{C_1\}$,

(4) $P_1 = \{C_1, C_3\}$, $P_2 = \{C_2\}$.

In general, given n chemicals and a container with x compartments. If all pair of chemicals are compatible, then there are x^n different valid assignments (x^n is the chromatic polynomial for this case). In Example (1) above, there are $x^n = 2^2 = 4$ different valid assignments.

On the other hand, if no pair of chemicals are compatible (any pair of chemicals are incompatible), then there are $x(x-1)(x-2)\cdots(x-n+1)$ different valid assignments (here

$x \geq n$). So $x(x-1)(x-2)\cdots(x-n+1)$ is the chromatic polynomial of this case. In Example (2) above, there are $x(x-1) = 2(2-1) = 2$ different valid assignments.

Similarly, the chromatic polynomial for Example (3) above is $x^3 - x^2$. Since in Example (3) $x = 2$ and there are $2^3 - 2^2 = 8 - 4 = 4$ different valid assignments.

Given n chemicals with their incompatible relations, please write a program to output the corresponding chromatic polynomial.

Input File Format

There are at most 10 test cases. The first line of each test case consists of an integer $N (2 \leq N \leq 50)$, where N is the number of chemicals the company has manufactured. The second line of each test case consists of an integer E $(1 \leq E \leq 100)$, where E is the number of incompatible pair of chemicals. Each of the following E lines consists of two integers C_i and C_j (separated by a space) $(1 \leq C_i \leq N, 1 \leq C_j \leq N)$, which means that C_i and C_j are incompatible chemicals. The last test case will be followed by a line containing a single 0 on a line.

Output Format

The output for each test case should contain a set of $N+1$ integers $\{a_N, a_{N-1}, \cdots, a_0\}$ (separated by a space) denoting the coefficients of the corresponding chromatic polynomial $a_N x^N + a_{N-1} x^{N-1} + \cdots + a_0$. Please output in the order of $a_N, a_{N-1}, ..., a_0$.

Sample Input

```
2
1
1 2
3
1
1 2
0
```

Output for the Sample Input

```
1 -1 0
1 -1 0 0
```

Problem K
Surveillance and Risk
Time Limit: *3 Seconds*

The mayor of X city wants to reduce the crimes on the streets. He decides to install surveillance cameras for the streets. There are m streets in X city. However, due to limited budget, the city can only install k cameras, and therefore possibly not all of the streets can be monitored. To make the surveillance system as effective as possible, the city needs to determine which streets should be monitored.

Each street has two endpoints. A street is *monitored* if a camera is installed at one of its endpoints. Otherwise, it is non-monitored. For each street, we are also given a risk value. Given the endpoints and the risk value of every streets, your job is to determine where the k cameras should be installed such that the maximal risk value among all *non-monitored* streets is minimized.

Technical Specification

1. The number of streets m is between 1 and 1000 for each test case.
2. The endpoints of streets are labelled by non-negative integers less than 1000.
3. The risk value of each street is a positive integer at most 256.
4. The number of cameras k is a positive integer at most 50 for each test case.

Input File Format
The first line of the input file contains an integer T which indicates the number of test cases. Each test case starts with two integers m and k in one line. Followed this line, there are m lines, one line for one street. Each line contains three integers u_i, v_i, and r_i, where u_i and v_i are the two endpoints of the street and r_i is its risk value. Two consecutive numbers in the same line are separated by a space.

Output Format
For each test case, output the minimized maximal risk of non-monitored streets on one line. If all streets can be monitored, output 0.

Sample Input

```
2
7 2
0 1 10
1 0 20
1 2 20
2 3 30
2 3 70
3 4 40
5 8 50
```

```
4 2
1 5 50
5 15 100
15 20 120
20 1 150
```

Output for the Sample Input

```
20
0
```

Problem L
The Maximum WN-Shape Sum
Time Limit: *3 seconds*

Given a matrix with R rows and C columns, where each element of the matrix is an integer, the problem is to compute the maximum sum for a particular shape called WN-Shape. The WN-Shape consists of two parts: W-shape and N-shape, with W-shape on the left and N-shape on the right. A shape is called a *W-shape* when its top contour inclines or remains horizontal (non-decreasing) and the bottom contour declines or remains horizontal (non-increasing) from left to right (see Figure 1). A mirror image of the W-shape is called an N-shape. A shape is called an *N-shape* when its top contour declines or remains horizontal and the bottom contour inclines or remains horizontal from left to right (see Figure 2).

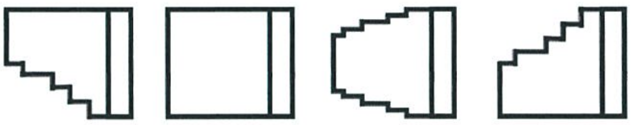


Figure 1: W-shapes.

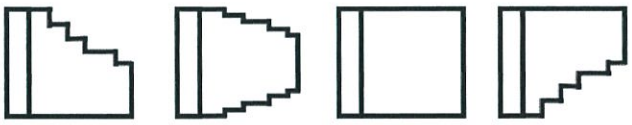


Figure 2: N-shapes.

As a result, there are $4 \times 4 = 16$ kinds of WN-Shapes in total. For example, you can combine a W-shape and an N-shape to form a WN-shape Figure 3. Note that the middle part must exactly overlap in order to fit one of 16 kinds of WN-Shapes. Also, note that all three kinds of shapes (W, N and WN-shape) must contain at least one element.

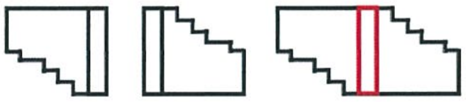


Figure 3: Combining a W-shape and an N-shape to form a WN-shape.

There are some illegal shapes that can not be classified to be WN-Shapes. Figure 4(a) is one of examples. If the middle part doesn't overlap properly, such as Figure 4(b), it will not be a legal one. To make it legal, you must change it a little bit as shown in Figure 4(c) or Figure 4(d).

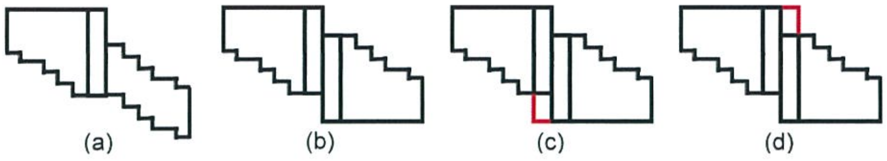
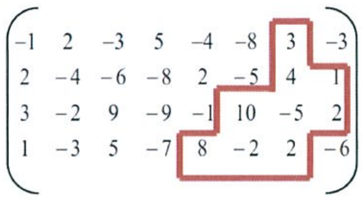


Figure 4: Some non-WN shapes.



$$\begin{pmatrix} -1 & 2 & -3 & 5 & -4 & -8 & 3 & -3 \\ 2 & -4 & -6 & -8 & 2 & -5 & 4 & 1 \\ 3 & -2 & 9 & -9 & -1 & 10 & -5 & 2 \\ 1 & -3 & 5 & -7 & 8 & -2 & 2 & -6 \end{pmatrix}$$

Figure 5: The maximum WN-shape sum equals 23.

A WN-shape sum is the sum of all elements in the WN-Shape. The maximum WN-shape sum is the maximum value among all the WN-shape sums in the given matrix. Figure 5 illustrates an example.

Given a matrix with R rows and C columns, where each entry of the matrix is an integer, your task is to write a computer program to compute the maximum WN-shape sum.

Technical Specification

- $1 \leq R \leq 100$
- $1 \leq C \leq 100$
- $-10^9 \leq$ the value of each element $\leq 10^9$

Input Format

The first line of the input contains an integer, denoting the number of test cases to follow. For each test case: the first line contains two positive integers R and C that are separated by a space. The next R lines represent R rows of the matrix such that each of the R lines contain C integers in which any two consecutive integers are separated by a space.

Output Format

For each test case, output the maximum WN-shape sum in one line.

Sample Input

```
2
4 8
-1 2 -3 5 -4 -8 3 -3
```

```
2 -4 -6 -8 2 -5 4 1
3 -2 9 -9 -1 10 -5 2
1 -3 5 -7 8 -2 2 -6
2 2
1 1
1 1
```

Output for the Sample Input

```
23
4
```