# 2019 National Collegiate Programming Contest
# Final Round

## October 19, 2019

- Problems: There are 15 tasks (28 pages in all) in this packet.
- Program Input: Input to the program are through standard input. Program input 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 the standard output (screen output).
- Time Limit: Judges will run each submitted program with certain time limit (given in the table below).

Table 1: **Task Information**

| | Task Name | Time Limit |
|---|---|---|
| Problem A | Factoring RSA Modulus | 1 sec. |
| Problem B | Wonderful Tours | 1 sec. |
| Problem C | Emergency Centers | 1 sec. |
| Problem D | Mutual Friends | 1 sec. |
| Problem E | People Counting | 3 sec. |
| Problem F | Coding a Sequence | 1 sec. |
| Problem G | Connected Sticks | 1 sec. |
| Problem H | Concyclic Points | 1 sec. |
| Problem I | Planning a Tour | 6 sec. |
| Problem J | Maintenance Schedule | 1 sec. |
| Problem K | Coin Game | 1 sec. |
| Problem L | MRT Routes | 2 sec. |
| Problem M | Rolling Dice | 1 sec. |
| Problem N | Storage Devices | 2 sec. |
| Problem O | Price is Right | 1 sec. |

# Problem A
## Factoring RSA Modulus

In a RSA cryptosystem, the *modulus* $n$ is a product of two large distinct primes $p$ and $q$. In general, there are no known efficient algorithms for factoring large composite integers. However, if $p$ and $q$ are not carefully selected, it is possible to factor $n$ efficiently. For example, if

$$2 < |p - q| < \sqrt[4]{n} \tag{1}$$

then the following algorithm can be used to factor $n$ efficiently.

Let $x = \lfloor \sqrt{n} \rfloor$, $p = x - a$, and $q = x + b$, where $a$ and $b$ are two positive integers. The values of $a$ and $b$ can be computed by the following method.

$$n = pq = (x - a)(x + b) = x^2 + (b - a)x - ab = x^2 + (b - a - 1)x + (x - ab). \tag{2}$$

If Inequality 1 holds, then

$$\begin{cases} 0 \le b - a - 1 < x, \\ 0 \le x - ab < x. \end{cases} \tag{3}$$

That is, $(1, b - a - 1, x - ab)_x$ is the $x$-radix representation of $n$. (See Equation 2 above.)

On the other hand, given $n$ and $x$, the $x$-radix representation of $n$, $(1, \alpha, \beta)$, can be computed easily. For example, $n = 221$, $x = \left\lfloor \sqrt{221} \right\rfloor = 14$, thus $221 = 1 \times 14^2 + 1 \times 14 + 11$.

It is known that the $x$-radix representation of a positive integer is unique. Therefore, we can factor $n$ by solving the following equations for $a$ and $b$.

$$(1, b - a - 1, x - ab) = (1, \alpha, \beta) \tag{4}$$

Write a program based on the method described above to factor RSA modulus. Note that if Inequality 1 holds then Inequality 3 must hold, which means that $n$ can be factored by using this method. This is because Equation 4 must have an integer solution. Obviously, the above method cannot factor all RSA moduli, because Equation 4 may have no integer solutions. In some cases, trying different values of $x$ may work. Therefore, if $x = \lfloor \sqrt{n} \rfloor$ fails, try to factor $n$ again by trying $x = \lfloor \sqrt{n} \rfloor - 1$. If that fails again, then try $x = \lfloor \sqrt{n} \rfloor + 1$. Note that, if $x^2 > n$, then at least one of $\alpha$ and $\beta$ must be negative. For simplicaty, in this problem, it is required that $x^2 \le n$. In other words, do not try to factor $n$ if $x^2 > n$.

# Input File Format

The first line of the input contains a positive inter $m$ ($m < 30$) indicating the number of test cases that follows. For teach test cawe, there is a positive integer $n$ in a line, where $2 < n < 2^{31}$. than $2^{31}$.

# Output Format

For each test integer $n$, print out the following in a line.

```
n = p * q
```

If the integer $n$ cannot be factored by $x = \lfloor\sqrt{n}\rfloor$, $x = \lfloor\sqrt{n}\rfloor - 1$, and $x = \lfloor\sqrt{n}\rfloor + 1$ then print

```
n = 1 * n
```

Print exactly 1 space before and after the symbols "=" and "*".

## Sample Input

```
3
221
187
33
```

## Output for the Sample Input

```
221 = 13 * 17
187 = 11 * 17
33 = 1 * 33
```

# Problem B
## Wonderful Tours

The ACM kingdom has $n$ cities, numbered from 0 to $n-1$, and $r$ unidirectional roads. Each unidirectional road starts at a city and ends at another. Sensibly, going along a unidirectional road $r_i$ incurs a cost, which we denote by $c(r_i)$. Because a unidirectional road $r_i$ may pass through sightseeing spot, its cost $c(r_i)$ may be negative. We want to know if it is possible to start from a city, traverse through several unidirectional roads and end up at the starting city, while incurring a negative cost in total. You may assume that every city is reachable from city 0. Furthermore, there is at most one unidirectional road from city $i$ to city $j$ for each pair $(i, j)$ of distinct cities.

# Technical Specification

1. $2 \leq n \leq 100$.
2. $|c(r_i)| \leq 50$ for all unidirectional road $r_i$.

# Input Format

The first line of input contains one integer $m$ ($m \leq 40$) denoting the number of test cases to follow. For each test case, the first two lines each contains a single integer, $n$ and $r$, respectively, denoting the number of cities and the number of unidirectional roads. The next $r$ lines each contains three integers, $i, j, c$, denoting that there is an unidirectional road from city $i$ to city $j$ with cost $c$.

# Output Format

For each test case, output "Wonderful Tour" if it is possible to start from a city, traverse through several unidirectional roads and end up at the starting city, while incurring a negative cost in total. Output "No Wonderful Tours" otherwise.

# Sample Input

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

```
1 0 -3
0 2 4
0 3 3
5
7
0 1 -6
0 4 5
0 1 2
0 2 1
3 0 -5
1 3 1
2 3 3
6
7
0 1 4
0 3 9
1 2 -1
3 2 2
3 4 -5
2 5 3
4 5 0
```

## Sample Output for the Sample Input

```
Wonderful Tour
Wonderful Tour
Wonderful Tour
No Wonderful Tours
```

# Problem C
## Emergency Centers

The government would like to bulid an emergency center on the freeway. There are many places at which the emergency center can be built. The cost for a place to be the emergency center relies on the distance from the place to the farthest location on the freeway. Precisely, let $P$ be the set of locations on the freeway and $d(x, y)$ be the distance between two locations $x$ and $y$. The cost of a location $x$ is defined as

$$\max\{d(x, y):\ y \in P\}.$$

Here we assume that $P$ is finite. The place(s) with the minimum cost is expected to be the emergency center. You are asked to help the government find the place(s) whose cost is minimum.

# Technical Specification

1. The set $P$ of locations on the freeway is exactly the set of places for building the emergency center. $P$ is finite, indexed from 0 to $p - 1$, and has at most $10^4$ elements.
2. The freeway forms a tree network, with vertex set $P$.
3. Distance between adjacent places are integers in $[1, 1000]$.
4. $d(x, y)$ is the length of the path running between $x$ and $y$.

# Input Format

The first line contains an integer $n$ ($1 \leq n \leq 12$) denoting the number of test cases to follow. For each test case, the first line contnains a single integer, $|P|$. Each of the following $|P| - 1$ lines consists of three numbers, $x$, $y$, and $d(x, y)$, where $x$ and $y$ are two adjacent places.

# Output Format

For each test case, output the place(s) with the minimum cost in ascending order.

# Sample Input

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

## Sample Output for the Sample Input

```
0 2
2
```

# Problem D
## Mutual Friends

Every year the X company recruits many new employees. To let the new members get familiar with their colleagues, the manager asks Alice, the leader of the human resource department, to organize some tutorials for them. She decides to divide the members into groups so that members in the same group are unfamiliar with each other. For each group she arranges a tutorial. However, the budget from the company is limited, and thus Alice looks for the least possible number of groups. When grouping the new employees, she found that among any three of them, at least two are (mutual) friends. Please help Alice find the minimum number of groups.

# Technical Specification

1. There are $n$ new employees, $3 \leq n \leq 500$.
2. There are $m$ pairs of employees that are mutual friends, $m \leq \binom{n}{2}$.
3. It is guaranteed that among any three of the employees, at least two are mutual friends.

# Input Format

The first line of input contains a single integer $x$ ($1 \leq x \leq 10$) denoting the number of test cases to follow. For each test case, the first line contains two integers, $n$ and $m$, denoting the number of new employees and the number of pair of mutual friendns, respectively. Each of the following $m$ lines consists of two integers $a$ and $b$, $0 \leq a, b \leq n-1$, denoting that employees $a$ and $b$ are mutual friends.

# Output Format

For each test case, output a positive integer which is the minimum number of groups.

# Sample Input

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

## Sample Output for the Sample Input

```
3
2
```

# Problem E
## People Counting

Tom, an employee of NCPC company that developes softwares for monitoring systems, encounters a problem as follows: given an area $A$ and a set of persons $P$, count the number of persons that are inside the area $A$. In this problem, $A$ is a polygon with interior angles of only 90 or 270 degrees. Consider the example in Figure 1. There are three persons $p_2$, $p_3$, and $p_5$ inside the area $A$. Note that persons on the border of $A$ are considered as inside. Please write a program to help Tom count the people inside the given area.

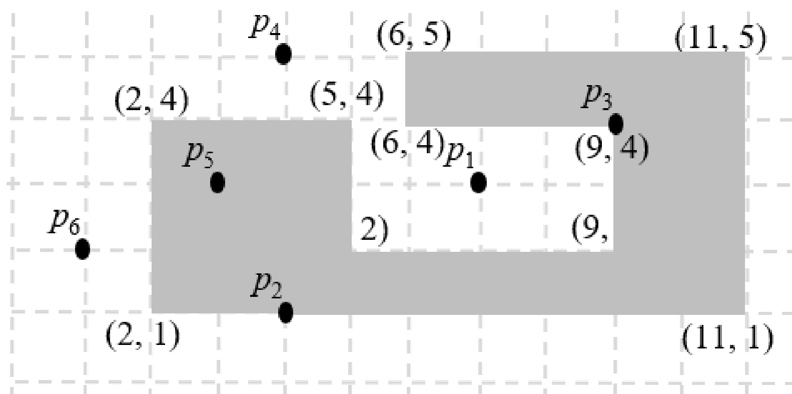


Figure 1: An area $A$ and a set $P = \{p_1, p_2, ..., p_6\}$ of persons.

## Technical Specification

1. The number $n$ of the vertices of $A$ is a positive integer between 4 and $10^6$.
2. The $x$- and $y$-coordinates of each vertex of $A$ are integers between 0 and $10^6$.
3. The number, $m$, of persons is an integer between 4 and $10^6$.
4. The $x$- and $y$-coordinates of each person are integers between 0 and $10^6$.

## Input Format

The first line of the input contains an integer $t$ ($t \leq 10$) indicating the number of test cases. The first line of each test case gives the 2 integers $n$ and $m$ ($4 \leq n, m \leq 10^6$), where $n$ is the number of vertices of $A$ and $m$ is the number of persons in $P$. Then, $n$ lines follow, each of which contains two integers $x$ and $y$, $0 \leq x, y \leq 10^6$, which are the integer coordinates of the vertices of the area $A$ in the order they would be visited in a trip counter-clockwise around the polygon. Then, $m$ lines follow, each of which contains two integers $x$ and $y$, $0 \leq x, y \leq 10^6$, which are the integer coordinates of the $m$ persons in $P$.

## Output Format

For each test case, output the number of persons inside the area $A$ in one line.

## Sample Input

```
2
10 6
11 1
11 5
6 5
6 4
9 4
9 2
5 2
5 4
2 4
2 1
7 3
4 2
9 4
4 5
3 3
1 2
4 1
2 1
5 1
5 4
2 4
1 2
```

## Sample Output for the Sample Input

```
3
0
```

# Problem F
## Coding a Sequence

Dr. Rissanen introduces a method to compress a string. This method uses a tag (a real number between 0 and 1) to represent a string. For example, suppose a string consists of letters from the alphabet $S = \{a, b, c\}$ with the probabilities $P(a) = 0.8$, $P(b) = 0.02$, and $P(c) = 0.18$, we first divide the unit length interval as shown in Figure 1.

The partition in which the tag resides depends on the first symbol of the input string. For example, if the first letter is $a$, the tag lies in the interval [0, 0.8); if the first letter is $b$, the tag lies in the interval [0.8, 0.82); and if the first letter is $c$, the tag lies in the interval [0.82, 1).

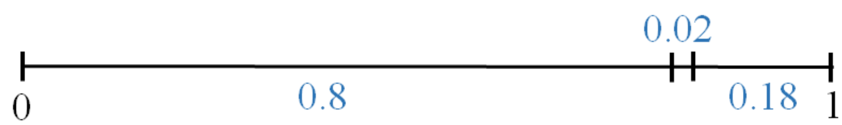


Figure 1: Illustrating the partition of an interval

Suppose now we have an input string '*acba*'. Because the first letter is $a$, the tag will be contained in the subinterval [0, 0.8). This subinterval is then divided in exactly the same proportions as the original interval, yielding the subintervals [0, 0.64), [0.64, 0.656) and [0.656, 0.8). The second letter is $c$; therefore, the tag will lie in [0.656, 0.8). The third letter is $b$, determining that the tag should lie in [0.7712, 0.77408). The last letter $a$ restricts the tag to lie between 0.7712 $(0.\mathbf{110001}0101101101011)_2$ and 0.7735 $(0.\mathbf{110001}1000000100001)_2$, where the binary sequences in parentheses are the binary representation of the numbers. Any number in [0.7712, 0.7735) can be used as the tag to represent the input string 'acba'. Figure 2 shows this example.

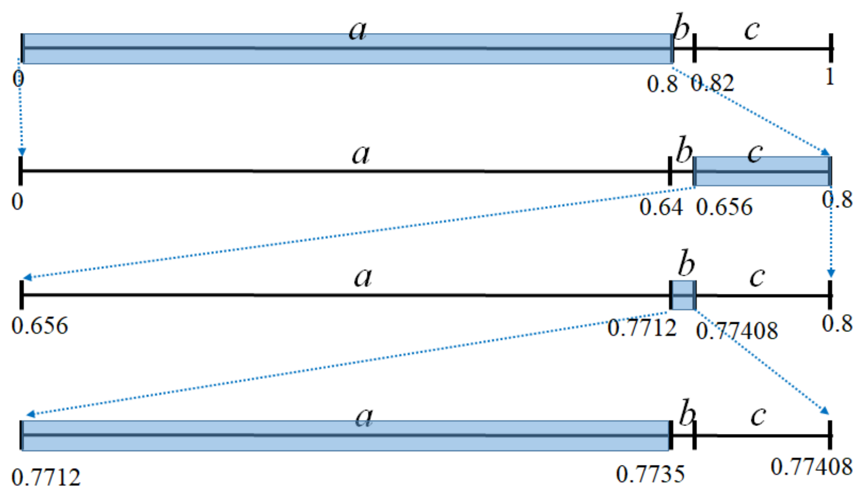


Figure 2: Illustrating coding procedure

Given the alphabet, the probability of each letter in the alphabet, and an input string, please compute the tag using Rissanen's method. Please note that some test cases involve the precision problem—a direct implementation of the algorithm may cause the endpoints of a subinterval to converge to the same point.

## Input Format

The first line contains a single integer $m$ ($m \leq 10$) denoting the number of test cases to follow. For each test case, there are four lines of input. The first line contains an integer $S$ ($1 \leq S \leq 26$), representing the alphabet size. The second line has $S$ distinct letters. The third line has $S$ real numbers $p_i$ ($0 < p_i < 1, 1 \leq i \leq S$), denoting the probability of each letter. $\sum_i p_i = 1$. The forth line contains a string with length $N$ ($1 \leq N \leq 2000$), representing the input string.

## Output Format

The output for each test case is a binary sequence, representing the compression result of the input string. Please return the *common* most significant bits (MSBs) of the final subinterval.

## Sample Input

```
2
3
abc
0.8 0.02 0.18
acba
3
abc
0.7 0.1 0.2
abcb
```

## Sample Output for the Sample Input

```
110001
10001110
```

# Problem G
## Connected Sticks

We have numerous sticks scattered on a table. The sticks may have different lengths: some are long and some are short. We are concerned about whether two sticks are connected by a sequence of touching (intersecting, or overlapping) sticks. In other words, two sticks can be directly touched, or indirectly connected via other touched sticks. Given a set of sticks, please compute the number of stick pairs that are connected.

## Input Format

The first line of input contains a single integer $m$ ($m \leq 10$) denoting the number of test cases to follow. For each test case, there are several lines of input. The first line contains an integer $N$ ($1 < N < 50$), representing the number of sticks. Each of the following $N$ lines contains four integers $x_1, y_1, x_2, y_2$ ($0 < x_1, y_1, x_2, y_2 < 100$), indicating the endpoints $(x_1, y_1), (x_2, y_2)$ for a stick. The length of a stick must be a positive number, i.e., $(x_1, y_1) \neq (x_2, y_2)$.

## Output Format

The output for each test case is the number of stick pairs that are connected.

## Sample Input

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

## Sample Output

```
0
3
```

# Problem H
## Concyclic Points

Alice draws three line segments in the plane (depicted in Figure 1) so that

(a) $\overline{EF}$ is perpendicular to $\overline{AB}$ and $\overline{CD}$, and

(b) $\overline{EF}$ intersects $\overline{AB}$ at point $X$ and intersects $\overline{CD}$ at point $Y$.

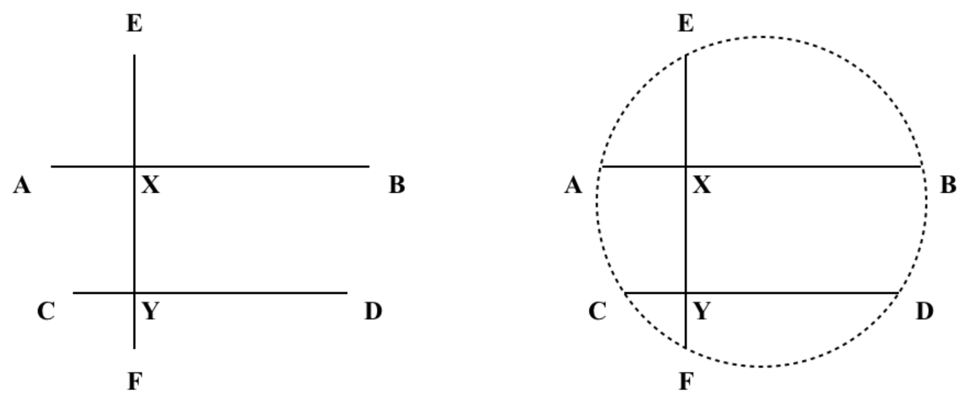


Figure 1: An illustration of Alice's drawing.

The exact configuration of Alice's drawing is given by specifying the length of the seven segments. Alice wonders whether the points $A, B, C, D, E, F$ are concyclic and asks you to write a program to assist her to figure it out. Note that a set of points are *concyclic* if there exists a circle that passes through all of them.

# Input Format

The first line of input contains an integer $t$ ($t \leq 40$) denotinng the number of test cases to follow. For each test case, there is a single line with 7 integers, denoting the length of the seven segments

$$\ell_{AX}, \ell_{XB}, \ell_{CY}, \ell_{YD}, \ell_{EX}, \ell_{XY}, \ell_{YF} \in \{1, 2, \ldots, 1000\},$$

respectively.

# Output Format

For each test case, output "Yes" if $A, B, C, D, E, F$ are concyclic, or "No" otherwise, on a single line.

# Sample Input

```
2
2 6 2 6 2 4 2
2 6 2 5 2 4 2
```

## Sample Output for the Sample Input

```
Yes
No
```

# Problem I
## Planninng a Tour

Alice is planning a tour of cities in Europe. Because of budget constraints, the tour needs to visit only four cities without passing through any other city. That is, on the route map, if each route is represented by an edge, then the tour is composed of edges

$$(c_1, c_2), (c_2, c_3), (c_3, c_4), \text{ and } (c_4, c_1)$$

for some four distinct cities $c_1, c_2, c_3, c_4$. We assume that each route is bidirectional, so route $(a, b)$ exists iff route $(b, a)$ exists. Consider that the number of cities in Europe is quite large, finding a feasible route may be difficult for Alice. Write a program to output a route that matches Alice's needs. If there are multiple feasible routes, output any of them.

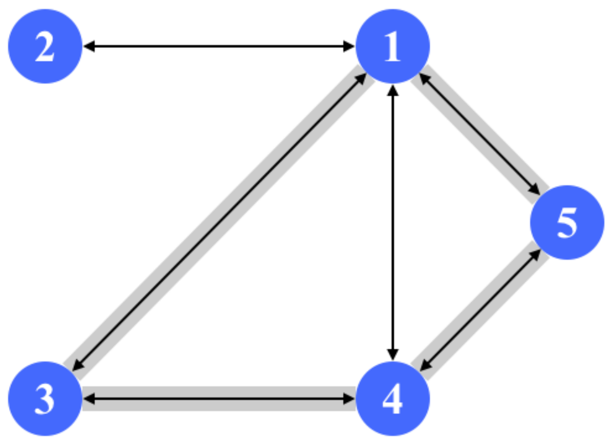


Figure 1: An illustration of a route map. A feasible route could be $3 \to 4 \to 5 \to 1 \to 3$. Note that there are eight different ways to represent a route of four distinct cities.

# Input Format

The first line of input contains an integer $t$ ($t \le 40$) denoting the number of test cases to follow. Each test case consists of two lines of input. The first line contains two integers $n$ and $m$, where $n$ denotes the number of cities in Europe and $m$ denotes the number of routes. Then, $m$ routes follow where each route is represented by a pair of integers $(u, v)$ for some $u \ne v \in \{1, 2, \ldots, n\}$ denoting a route from city $u$ to city $v$. It is guaranteed that $n \le 10^4$ and $m \le \min\{n(n-1)/2, 10^6\}$.

# Output Format

For each test case, if there is a feasible tour that matches Alice's needs, output the four cites of the feasible tour in order; otherwise output $-1$.

## Sample Input

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

## Sample Output for the Sample Input

```
1 3 4 2
-1
```

# Problem J
## Maintenance Schedule

Bob is the manager of the City Metro Maintenance Unit. His main duty is ensure the reliability and stability of the Metro service. Therefore, routine inspection and testing of the railway system without affecting the daily operation is crucial. The Metro system consists of $n$ stations and $m$ links connecting the stations. For simplicity, Bob models it as an undirected graph $G(V, E)$ with $V = \{1, 2, \ldots, n\}$ and $E = \{(i, j) : \text{if there is a railway between station } i \text{ and } j\}$. Due to the urban planning, each station has at most four links passing through it. According to the condition of the railways, Bob schedules to regularly exam each link $(i, j) \in E$ for $t_{i,j}$ times. Because of the shortage of crew staff, every day Bob can only arrange enough workforce to check $r_i$ links incident at station $i$ for each $i \in V$, where $r_i$ and $t_{i,j}$ are all positive integers. Note a link may be inspected multiple times a day. To secure the budget for this project, Bob wants to know the minimum number of day(s) required to accomplish the yearly routine maintenance.

Your task is to write a program to help Bob find the minimum number of working days .

# Technical Specification

1. $1 < n < 1000$
2. $0 < r_i, t_{i,j} < 2^{32}$

# Input Format

The first line of the input gives the number of test cases, $K$ ($K < 10$). For each case, the first line consists of two positive integers $n$ and $m$, separated by space(s), indicating the number of stations and links, respectively. Then $n$ lines follow, where the $i$-th ($i = 1, \ldots, n$) line has a positive integer $r_i$ indicating the number of links incident at station $i$ that can be checked in a day. Then $m$ lines follow, where the $k$-th ($k = 1, \ldots, m$) line has 3 positive integer $a, b, t$, indicating the link between stations $a$ and $b$ needs to be inspected $t$ times every year.

# Output Format

For each test case, output one line that contains the minimum required day(s) for maintenance.

## Sample Input

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

## Sample Output for the Sample Input

```
2
2
```

# Problem K
## Coins Game

Iron Man and Captain America play a game with piles of coins. There are $n$ ($n$ is *even*) number of coin piles, *arranged in a row*. Each pile has at least one coin. Denote the number of coins of pile $i$ as $C_i$.

The objective of the game is to end with *the most* coins. Since the total number of coins is *odd*, there can be no ties.

Iron Man and Captain America take turns, with *Iron Man starting first*. At each turn, a player takes the entire pile of coins from either the beginning or the end of the row of piles of coin. This continues until there are no more piles left, at which point the person with the most coins wins.

Assuming Iron Man and Captain America play *optimally*, return **the maximum difference of Iron Man's and Captain America's coins** at the end of the game.

# Technical Specification

1. $2 \leq n \leq 500$, $n$ is *even*.
2. $1 \leq C_i \leq 500$ for all $i$.
3. Sum of all coins in all piles is *odd*.

# Input Format

The first line of the input contains an integer $k$ ($k \leq 10$), indicating the number of test cases to follow. For each test case, there is a single line with several integers. The first integer is $n$, followed by $n$ integers, $C_1, C_2, ..., C_n$, which are the number of coins in pile 1, 2, ..., n, respectively.

# Output Format

Assuming Iron Man and Captain America play optimally, output **maximum difference of Iron Man's and Captain America's coins**.

# Sample Input

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

# Sample Output for the Sample Input

```
1
```

# Problem L
## MRT Routes

NCPC city has a list of MRT routes. Train on each route make stops at a number of stations. The MRT trains are fully automated so the trains for each route all run in loops. For example, if route $i$ stops at stations 1, 5, 7, then the MRT trains will run continuously and make stops at stations 1, 5, 7, 1, 5, 7, .... If we are at MRT station $S$ (waiting for the next train), and wanting to go to MRT station $T$, travelling by MRTs only. What is the least number of MRT routes we must take to reach our destination? Return -1 if it is not possible.

# Technical Specification

1. $1 \leq R$, total number of routes $\leq 500$.
2. $1 \leq R_i$, total number of stations on route $i \leq 500$.
3. $0 \leq$ total number of distinct MRT stations $< 10^6$. Stations are numbers from 1 to $10^6 - 1$.

# Input Format

The first line of the input contains an integer $m$ denoting the number of test cases. For each test case, the first line contains an integer denoting the starting station $S$. The second line also contains an integer denoting the terminal station $T$. The third line contains an integer $R$, denoting the total number of routes. For the next $k$ lines, the first integer $R_i$ is the number stations on route $i$, followed by $R_i$ integers denoting the station number of the stops (in order) along route $i$.

# Output Format

Output the least number of MRT routes we must take to reach destination station. Output "-1" if it is not possible to reach station $T$ from station $S$.

# Sample Input

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

## Sample Output for the Sample Input

```
2
```

# Problem M
## Rolling Dice

A die is a cubic object with six faces, there is a number between 0 and 9 on each face. A manufacture produces many dice with different numbers on each face. For example, a die that sits on a table can be numbered (1,2,3,3,2,1) in the north-, east-, south-, west-, top-, bottom-facing faces, respectively. The manufacture holds a game to promote sale of these dice. First, a die $A$ is show on a table. Of course, only the numbers on the five visible faces can be seen. Second, there is another die $B$ on each participant's table. Each participant is asked to roll die $B$ so as to show the same numbers and directions as with die $A$ if possible.

For simplicity, die $B$ can only be rolled on the table in four directions, that is, north, east, south, and west. For example, while rolling a die to the east, we mean to rotate the die by the axis from north to south and turn the top side toward the east. Also, a configuration of a die is a sequence of numbers on six faces (north, east, south, west, top, bottom).

Given two dice $A$ and $B$, determine the minimum number of rolling action required to transform die $B$ to look like die $A$ on the five visible faces on each die. For example, die $A$ is represented as five numbers (2,2,1,1,1) on five faces (north, east, south, west, top). Die $B$ is configured as six numbers (2,2,1,1,2,1) on six faces (north, east, south, west, top, bottom). If we roll die $B$ in "north" direction once, die $B$ would show (2,2,1,1,1,-) which is the same with die $A$ on the five visible faces. So only one action is for the transformation.

# Input Format

The first line contains a integer $t$ ($t \leq 10$) denoting the number of test cases to follow. Each test case consists of two lines of input The first line has five numbers that are on the visible faces of die $A$. The second line has six numbers that are on the faces of the die $B$.

# Output Format

For each test case, output the minimum number of actions to apply on die $B$ such that the final result will have die $B$ looking just like dice $A$ (on the five visible faces.) If it is not possible to make die $B$ looking like die $A$, output $-1$.

# Sample Input

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

# Sample Output for the Sample Input

```
0
```

```
1
-1
```

# Problem N
## Storage Devices

You have a mass storage system with a number of concurrently running processes. Each of the processes deals with a pair of storage devices at a time. To prevent concurrent access from different processes to a storage device, the storage devices are protected by mutual exclusion protocols which are also known as mutex-locks. It is guaranteed that no two processes can hold the same lock at a time, and the access to a storage device is allowed only when the process possesses the lock for that device.

The typical behavior of a process that deals with storage devices P and Q goes as follows:

```
loop forever

    P.lock()
    Q.lock()

    Work with P and Q.

    Q.unlock()
    P.unlock()

end loop
```

The order in which the locks for devices P and Q are taken is important. Consider the following scenario where a process, say, $p_a$, is holding the lock for a device P and is waiting for the lock for device Q that is currently taken by another process $p_b$. Apparently, process $p_a$ cannot proceed unless it receives the lock for Q after $p_b$ releases it. This creates a waiting chain of length 1, denoted $p_a \to p_b$, which means that $p_a$ is waiting for a lock from $p_b$.

The waiting chain, starting from $p_a$, can be longer if process $p_b$ is also waiting for a lock that is currently possessed by some other process. If the waiting chain eventually loops back to $p_a$ itself, then we say that the waiting chain has an infinite length and the system is in deadlock.

The order for which each process acquires the locks it needs decides whether or not the system could result in deadlock and also the maximum length of any waiting chain in the system.

Given the number of storage devices $N$, the number of processes $M$ in the system, and the pair of storage devices each process deals with, your task is to decide the order for which each process should acquire its locks so that, the system never deadlocks and the maximum length of any possible waiting chain is minimized.

# Technical Specification

- $2 \leq N \leq 15$
- $1 \leq M \leq 100$

# Input Format

The first line of input contains a integer $t$ ($t \leq 65$) denoting the number of test cases to follow.

Each test case starts with a line contains two integers $N$ and $M$.
Then there are $M$ lines, one for each process. Each line contains two integers $u$ and $v$, where $1 \leq u, v \leq N$, that describe the pair of storage devices each process deals with.

# Output Format

Output two lines for each test case. In the first line print the minimum length of the maximum waiting chain. In the second line print the order of acquiring locks each process should use. Print the answer for each process according to their input order. Use integer 0 to indicate that it should acquire $u$ first and use integer 1 to indicate otherwise.

If there are multiple solutions, print any of them.

# Sample Input

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

# Sample Output for the Sample Input

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

# Problem O
## Price is Right

Many people buy things online. They can easily find different prices for the same object from different online stores. For example, an 'ifoneX' may be sold at the price of 30000 NTD on the Pingo online store, but it may cost only 29999 NTD on the CPHome online store. Bob, an online shopping addict, who collects a list of items as well as many different online prices for each item. His list is too long to be processed by hands so he decides to find a program to help him determine the lowest price of each item.

## Technical Specification

1. The name of each item is composed of English letters and has at most 20 characters.
2. All prices are positive integers no more than $10^6$.
3. There are at most 20 different prices listed for an item.

## Input Format

The first line of the input contains a single integer $t$ ($t \leq 100$) denoting the number of test cases to follow. Each test case occupies one line. It starts with the item name followed by at least one price for that item. There can be several different prices for an item. The item name and the nearby associated prices are delimited by a space.

## Output Format

For each test case, output its item name and the lowest price for this item in one line. There should be a space between the name and the price.

## Sample Input

```
3
ifoneX 30000 29999
kindle 8999 9023 8900
RavFour 899999
```

## Sample Output for the Sample Input

```
ifoneX 29999
kindle 8900
RavFour 899999
```