A – The Third Law of Thermodynamics

Description

According to the laws of thermodynamics, eventually all particles in the universe will have the same energy. This means that in a very distant future the temperature of any two points in the whole universe will be the same and nothing moves anymore!
After watching a Youtube video about this, Laurent Lavoisier is curious about this phenomenon and wants to test its truth. Laurent starts with n integers (an extreme simplification of the particles in the universe). He then takes the minimum and maximum numbers and replaces both with their difference (i.e. if a is the maximum and b the minimum, then Laurent replaces both with $a - b$). He repeats the same operation over and over, until all numbers are equal.
He wants to know the final temperature of all particles when all numbers are equal, or if the video he watched is not factually correct (all numbers will never be equal), he wants his program to print "`Nope!`".

Input

The first line contains the number of test cases T ($1 \leq T \leq 32$).
Each test is consist of two lines. First, an integer n ($2 \leq n \leq 10$), and the second line n whitespace-separated integers x_i ($-100000 \leq x_i \leq 100000$).

Output

For each test case, if the task can be done, print one of the n integers after the task is completed, otherwise if it cannot be done, print "`Nope!`".

Sample Input

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

Sample Output

```
2
5
```

B – **LCS**

Description

According to Wikipedia:

> The longest common subsequence (LCS) problem is the problem of finding the longest subsequence common to all sequences in a set of sequences (often just two sequences). It differs from the longest common substring problem: unlike substrings, subsequences are not required to occupy consecutive positions within the original sequences. The longest common subsequence problem is a classic computer science problem, the basis of data comparison programs such as the diff utility, and has applications in bioinformatics. It is also widely used by revision control systems such as Git for reconciling multiple changes made to a revision-controlled collection of files.

In simpler terms, for two sequence of elements, if you can make the two sequences the same by deleting some elements from them, the remaining sequence is called "common sub-sequence". And the longest common sub-sequence of the two sequences, obviously, is called "LCS".
Now, you are given two sequences of integers and you can permute the elements of each sequence. Out of all the permutations, what is the maximum LCS of the two sequences?

Input

The first line contains the number of test cases T $(1 \leq T \leq 64)$.
For each test, the first line contains two integers M $(0 < M < 10000)$ and N $(0 < N < 10000)$, the length of each sequence. Next two lines, contain M and N integers respectively, which are the numbers in the two sequences.
The elements of the sequences are 32-bit signed integers.

Output

For each test, print one integer in one line, the maximum length of the LCS of all the arrangements of the two sequences.

Sample Input

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

Sample Output

```
2
3
```

C – **From My Moon to the Titan's Moons**

Description

There's a competition, sponsored by a company named XpaceS in planet Titan which has N moons. To enter the competition any participant has to complete the first task which is to put their team flag on all the moons in a specific order and land on the first moon they have started the task on, using the fuel available on the moons.
The order is circular so for $N = 3$, they can put the flag in the order M_0, M_1, M_2, M_0 or M_1, M_2, M_0, M_1 or M_2, M_0, M_1, M_2. It's up to the participants to choose their starting moon. The only restriction is that the spaceship they want to use to carry the flags should not have any fuel in the beginning and they should only use the fuel that is available on each moon.
Each moon i has a supply of F_i units of fuel that the spaceship can use to refuel once landed on the moon M_i. The spaceship's fuel tank is so large that it practically has infinite capacity.
If you want to finish this task, you need to be clever about choosing your starting moon. So write a program to find it.

Input

The first line contains the number of test cases T $(1 \leq T \leq 256)$.
Each test case begins with one integer N $(2 \leq N \leq 10000)$ the number of moons, then N lines will follow, each contains two non-negative integers F_i, the amount of fuel, and D_i, the distance to the next moon. D_{n-1} is the distance from the last moon to the first one.
The spaceship consumes one unit of fuel per one unit of distance.

Output

For each test, print one integer in one line, the ID number of the moon that could be the starting point. The ID of the first moon is 0, next is 1, and so on. If there are more than one possible starting moons, output the one that comes first in the input. If there is no such starting moon to finish the task, just output "`it's a trap`".

Sample Input

```
3
3
100 100
100 100
100 100
2
100 200
200 100
2
100 200
100 200
```

Sample Output

```
0
1
it's a trap
```

D – The Masochist

Description

For an integer x, $bud(x)$ is defined to be the largest divisor of x other than x itself. A *formation* of a given number x is a sequence $(x_1, ..., x_m)$, such that $x_1 + x_2 + ... + x_m = x$, m is an integer greater than 0, and for each $i \in \{1, ..., m\}$, $x_i > 1$ and x_i is an integer.

The *desirable* formation of x is a formation $(x_1, ..., x_m)$, in which $\sum_{i\in\{1,...,m\}} bud(x_i)$ is the minimum among all possible formations of x. The *desirable* number of x is $\sum_{i\in\{1,...,m\}} bud(x_i)$, such that $(x_1, ..., x_m)$ is the desirable formation of x. The *desirable* number of a set of numbers is defined as the summation of the desirable numbers of its elements.

Given two integer numbers N and K find a subset of size K from $\{y|2 <= y <= N\}$ that has the smallest desirable number.

Input

The first line of the input contains a single integer T ($1 \leq T \leq 50$), which is the number of test cases. Each of the following T lines contains two space-separated integers N and K ($2 \leq N \leq 10^7, 1 \leq K \leq N - 1$).

Output

For each test case print a single number in a separate line. It represents the minimum desirable number among all subsets of size K of $\{y|2 \leq y \leq N\}$.

Sample Input

```
3
10 5
100 15
9999 4321
```

Sample Output

```
6
15
7413
```

E – **Artam Clancy**

Description

Game design has changed a lot during the past few years. Nowadays games have many storylines that are executed based on player's choices. This paradigm not only makes the games more interesting but also allows fans to play the game several times and go through all storylines. Artam is writing the story of a new game. He has developed n levels in total, such that there is only one way to reach each level, and going from one level to the next takes exactly one hour. Also, when you finish a level you cannot go back to that level again without starting the game from level 1. Since this is a new game, the storyline might branch out in some levels. Now, Artam wants to create a new path (a shortcut) in the storyline, that takes the player from level 1 (start of the game) to some other level. His goal is to minimize the number of hours players spend on going through all levels of the game. Note that going through all levels of the game means starting at level 1 and finishing the game (going to the end of a branch in the storyline), then going back to level one and finishing the game in a different branch (different branchs are different in at least one level) in the storyline, and repeating this process until all possible endgames are reached. What is the minimum time a player can go through all levels with such shortcut?

Input

The first line contains the number of test cases T.
For each test, the first line contains an integer n ($1 \leq n \leq 2 \times 10^5$), the number of levels in the game. Followed by $n-1$ lines, each contains two integers u and v ($1 \leq u, v \leq n$), which means it takes one hour to go from level u to level v.
You cannot reach any level from two different levels, and the sum of all levels in all test cases does not exceed 5×10^5. Also the maximum time to finish the game once is limited to 3×10^4.
Use fast IO functions to read the input.

Output

For each test case, print a single integer in a line, the minimum number of hours required to go through all levels after adding the shortcut.

Sample Input

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

Sample Output

```
8
2
```

F – **Planet Squanch**

Description

Forget Mars, scientists have discovered a new Earth-sized, possibly habitable, planet just 11 light years away. Known as *Squanch*, the newly discovered planet orbits a life-friendly red dwarf star that is an estimated seven billion years old.

After discovering *Squanch*, space agencies started competing to find places on the planet that support life. NASA and the Russian Space Agency, both have sent their own designed robots to investigate the planet and find an area of the surface with a better chance at hosting an atmosphere that could support life. These two robots have different algorithms to find the habitable area, so they could possibly report different areas as habitable. And of course, these two areas could share some areas of the surface. Since both robots are reliable, reported area by each of them would support life.

Since *Squanch* is big enough, we can consider the surface to be a giant plane instead of a curved surface, so the area that a robot reports, forms a polygon on a plane. Each robot will send signals to its own agency and report the x-y positions of vertices of the polygon they've found. The origin point $(0, 0)$ for both robots is the same. Help scientist to find the total area on *Squanch* that supports life.

Input

The first line of input contains a single integer $T(1 \leq T \leq 64)$, which is the number of test cases.
For each test case, the first line contains two integers $3 \leq n, m \leq 50$, the number of vertices of the two polygons. The next n lines each contain two real numbers (x_i, y_i) (distances are in kilometers), the coordinates of the vertices of the first polygon, followed by m lines with the same format for the vertices of the second polygon.

Output

For each test case, print the area in km^2 that can support life, rounded up to 3 decimal places.

Sample Input

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

Sample Output

```
1.500
```

G – **KhanBaG's Mafia**

Description

A knockout tournament is about to be held in the town of Mooliland. This tournament has 2^k participants to whom we give a numbering of $1, 2, \ldots, 2^k$.

The tournament is held in k rounds. In the first round participants with numbers in the form of $2t+1$ and $2t+2$ compete and the winner goes to the next round. In the second round the winner of the match-up between $4t+1$ and $4t+2$ competes against the winner of the match-up between $4t+3$ and $4t+4$. In the same manner, in round i, the winner of the match-up between 2^it+1 and 2^it+2^{i-1} competes the winner of the match-up between $2^it+2^{i-1}+1$ and 2^it+2^i.

The participants' weights at the time of registration has been $w_1, w_2, \ldots, w_{2^k}$. KhanBaG likes the number W and has placed a bet claiming that the tournament champion's weight is going to be W. She is willing to do anything for the champion's weight to be equal to W. She has somehow found out that the participants who weigh less than their opponents win in the even rounds and those who weigh more than their opponent win in the odd rounds. Also, if the two opponents' weighs are equal, the winner is chosen randomly.

There's a clause in the tournament rules obligating the weight of the participants on competition day to be equal to their weight on registration day. KhanBaG can pay the administrators of the tournament an amount of X moollars as a bribe in order to change the clause to "the weight of the participants on competition day can differ at most X kilograms from their weight on registration day". She wants to win the bet by changing the rule and convincing several participants to change their weight, but she can ask at most m people for weight change due to security reasons. (All tournament rounds are held on the same day and no one's weight changes during that day.) What is the minimum bribe she has to pay in order to win the bet?

Input

In the first line of the input comes T, the number of test cases.

The first line of each test case contains k, W, and m, which represent the the number of rounds, KhanBaG's favorite number, and the maximum number of people she can ask to change their weights, respectively $1 \leq m \leq 2^k$, $1 \leq W \leq 10^9$ and $1 \leq k \leq 18$.

The second line of each test case contains 2^k numbers $w_1, w_2, \ldots, w_{2^k}$ representing the participants' weights on registration day.

Output

Print an integer representing the least amount of money KhanBaG should pay as a bribe to the administrators in order to win the bet. Print out -1 if she cannot win under any conditions.

Sample Input

```
2
2 15 3
50 30 20 40
2 20 1
10 10 10 10
```

Sample Output

```
25
-1
```

H – SimiN and NiwiS

Description

Simin loves french fries! She places n packs of french fries in a line on the table and then places a ketchup packet on top of each one. There are f types of french fries in the mix, represented with $1, 2, 3, \ldots, f$. Additionally, the ketchup packets also come in s flavors represented with $1, 2, 3, \ldots, s$. Simin prefers only certain ketchup flavors for each type of french fries. Niwis wants to rearrange the ketchup packets so that eventually there is still a ketchup packet on every french fries pack. Niwis would like to know if she changes the placement of the ketchup packets what is the minimum and the maximum number of packs of french fries that will go with the ketchup flavor Simin likes?

Input

In the first line of the input comes T, the number of test cases.
The first line of every test case consists of integers n, f and s respectively, in which $1 \leq n \leq 10^5$, $1 \leq f$ and $s \leq 300$.
The second line consists of n integers f_1, f_2, ..., f_n which are the types of french fries on the table. $(1 \leq f_i \leq f)$
The third line consists of n integers s_1, s_2, ..., s_n which are different ketchup flavors.$(1 \leq s_i \leq s)$
The following f lines, denote the rows of matrix $M_{f \times s}$, with each line containing s integers describing the columns. The element $M_{i,j}$ is 1 if and only if Simin likes the french fries of type i with the ketchup flavor of type j, otherwise it is 0.

Output

Considering all possible arrangements of the ketchup and french fries packets, print the minimum and the maximum number of pairs of french fries packs and ketchup flavors that Simin likes.

Sample Input

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

Sample Output

```
0 2
```

I – Moein is a Cheater

Description

Moein has done all the cheating he could to eventually reach the ACM-ICPC World Finals. In addition to his team, N other teams, represented with $1, 2, 3, \ldots, N$, have participated in the contest and some of those teams are friends with each other. On the opening day, he intends to have a chat with the other teams and share the ideas of the newest methods of cheating. He starts the first conversation with an arbitrary team. However, starting to chat with a team about cheating is somewhat difficult; therefore, after finishing his chat with team T, he asks them to introduce him to the next team. Team T accepts to do so only if they are friends with team S via at most K intermediaries. (e.g. If T and S are friends, they're friends via zero intermediaries, but if they're not friends with each other and have a mutual friend, they are friends via one intermediary.)

Moein is seeking the longest sequence of chats such that he chats with each team at most once. Help him find this sequence.

Input

The first line contains the number of test cases T.
For each test, the first line contains three integers $N(3 \leq N \leq 1000), K(2 \leq K \leq N)$ and $M(1 \leq M \leq 10^5)$ the number of friendships between teams. After that there are M lines each containing 2 integers x and y, that show x and y are friends. $1 \leq x < y \leq N$

Output

In the first line of output print the length of longest sequence, and print the sequence in the next line. If there are more than one sequence with the longest length, print one of them arbitrarily.

Sample Input

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

Sample Output

```
4
1 2 3 4
```

J – SH4R3C0D3

Description

In a galaxy far far away, a war is happening between planet SH4R3 and planet C0D3. The military forces of SH4R3 need your help to hack into C0D3's mother-ship communication system and inject some data that turns off all of the N battleships that C0D3 military force controls. Each battleship has a 7-character long code, with distinct characters, that turns off the engine if seen in a received message from the mother-ship. For example if the code is "destroy" for a battleship, it'll turn off the engine once it receives a message that contains that string, like "abcdestroydefg".
We have all the turn off codes for all the battleships, and we've already hacked into C0D3's mother-ship broadcast system. We just need to construct the string payload and broadcast to all the battleships. Since sending a character through our hacked link is expensive, we want to minimize the length of the payload. The security system of the battleship B_i, prevents the engine to turn off unless B_{i-1} is already turned off. So we need a strategic plan to turn off the battleships in order from 1 to N.
But that's not the only security system the battleships have in place. The firmware on the C0D3's battleships has a basic algorithm that detects if a message is a spam and should be thrown away, or if it's a legitimate message and should be executed. The algorithm is simple, a legitimate message is always a palindrome.
Now it's your job to find the shortest palindrome message that turns off all the battleships in order.

Input

The first line of input contains a single integer T ($1 \leq T \leq 128$), which is the number of test cases.
Each test begins with an integer N, the number of battleships. The next line contains N 7-character long turnoff commands, for battleship 1 through N.

Output

For each test case, print the payload that we need to broadcast through mother-ship disable battleships and win the war.
If multiple possible solution exists, print the one that is lexicographically smaller.
Each answer is guaranteed to be less than 100 characters long.

Sample Input

```
2
2
destroy destroy
3
shareck areckod okcerah
```

Sample Output

```
destroyortsedestroyortsed
shareckodokcerahs
```

K – **ICPC**

Description

Each year the Irregular Creative Pool game Consortium (ICPC) invents a game using pool billiards props and invites all the ICPC members from all over the world to participate and possibly win a plastic trophy. This year, the game is simple. We have N different balls, numbered from 1 to N. All the balls are in a row in front of the player. The player has to choose one of the balls - called cue ball - and shoot it at the other balls. The cue ball must hit exactly one of the balls - called target ball - on the table (otherwise, the player loses). After the hit, the target ball gets removed from the table, all the balls on the table get rearranged to form a row in front of the table again, and the player continues to choose another ball from table and shoots until there's only one ball left on the table. The player scores point cumulatively, based on which cue ball hits which target ball.
Assuming a cue ball always hits the target ball and the player will never lose, what is the maximum point a player can get, given the number of balls and the scoring matrix?

Input

The first line of input contains a single integer $T (1 \leq T \leq 512)$, which is the number of test cases.
For each test case, the first line contains integer $N (2 \leq N \leq 10)$, the number of balls. The next N lines each contain N integers which the j^{th} integer on the i^{th} line is the score for hitting A_j as the target ball with A_i as the cue ball $(0 < A_{ij} \leq 10000)$.

Output

For each test case, print the maximum points a player can obtain.

Sample Input

```
2
2
0 1
10 0
3
0 100 1
60 0 1
1 50 0
```

Sample Output

```
10
110
```

L – Another Brick in the Wall

Description

The Russia fiasco has finally caught up to Donald Trump and led to his impeachment. Looking for a job that suits an incompetent manager, he has landed himself a position as the head of the ECE department at the University of Tehran. The first item on his agenda is building a wall to keep undesirable students from other, lesser universities out. He has enlisted your help in accomplishing this task. Donald has an $a \times b$ wall in mind and you have an unlimited supply of 1×2 sized bricks at your disposal to build it.

Donald wants the wall to be strong; a wall is strong if and only if every line that goes between two consecutive rows or columns crosses at least one brick, i.e. if there is a line going between two successive rows or columns that only goes through the mortar between the bricks the wall is not strong.

For example the following wall is not strong because of the highlighted line:

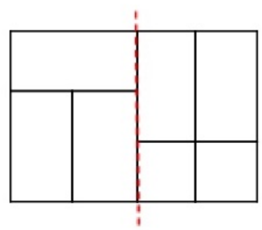

Figure 1: A 3×4 wall that is not strong.

Input

The first line of the input contains a single number $T \leq 100$ denoting the number of test cases. Each of the T following lines consists of two numbers $4 < a, b \leq 3000$ denoting the size of the wall.

Output

For every test case your program must output an $a \times b$ table of integers indicating the placement of the bricks in the wall. Every brick is identified by a number between zero (inclusive) and the total number of bricks used (exclusive). If building a strong wall of the given size is impossible, output a single line containing the string "`Impossible`".

Sample Input

```
2
5 6
5 5
```

Sample Output

```
0 1 1 2 3 3
0 4 4 2 5 6
7 8 9 9 5 6
7 8 11 12 12 14
10 10 11 13 13 14
Impossible
```