The 2024 ICPC Asia Topi Regional Final On-site Contest

Instructions

- Do not open the booklet unless you are explicitly told to do so. You can only read these instructions below.
- If you have any question regarding the problems, seek a clarification from the judges using DOMJudge.
- Before submitting a run, make sure that it is executable via command line. For Java, it must be executable via "javac" and for GNU C++ via "g++". Java programmers need to remove any "package" statements and source code's file name must be the same as of main class. C++ programmers need to remove any getch() / system("pause") like statements.
- Do not attach input files while submitting a run, only submit/attach source code files, i.e., *.java or *.cpp or *.py.
- Language supported: C/C++, Java and Python3
- Source code file name should not contain white space or special characters.
- You must take input from Console i.e.: Standard Input Stream (stdin in C, cin in C++, System.in in Java, stdin in Python)
- You must print your output to Console i.e.: Standard Output Stream (stdout in C, cout in C++, System.out in Java)
- Please, don't create/open any file for input or output.
- Please strictly meet the output format requirements as described in problem statements, because your program will be auto judged by computer. Your output will be compared with judge's output byte-by-byte and not tolerate even a difference of single byte. So, be aware! **Pay special attention to spaces, commas, dots, newlines, decimal places, case sensitivity etc.**
- All your programs must meet the time constraint specified.
- The decision of judges will be absolutely final.

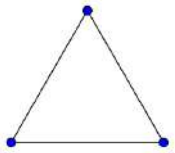
Problem 01: Circuit Design

Time limit: 18 seconds

Imagine you are an electronic engineer working on optimizing digital circuit layouts for integrated chips. Each circuit is represented as a graph where nodes correspond to components (like logic gates) and edges represent the connections between them.

Your task is to identify how often a specific subcircuit also called query subcircuit (a small group of components with a defined topology) appears in the overall circuit. Subcircuits can overlap, sharing components or connections, but each occurrence must have at least one unique component.

Given an undirected graph representing the circuit (components as nodes and connections as edges), identify and count all occurrences of the following pattern within the larger graph.



Input:

The input consists of the following:

- First line contains a pair ***N***, ***M***. The number of components in the circuit, ***N (1 ≤ N ≤ 2000)***. The number of edges in the circuit, ***M (0 ≤ M ≤ (N(N-1)/2))***.
- M lines, each containing two component IDs ***u*** and ***v (0 ≤ u, v ≤ N-1)***, representing an undirected edge from component ***u*** to component ***v*** in the circuit.

Output:

Output a single integer, the count of occurrences of the query sub circuit within the provided overall circuit.

Sample Input	**Sample Output**
6 9 0 1 1 2 1 3 2 3 3 4 4 0 2 5 5 0 5 1	3

Problem 02: Car Washing

Time limit: 1 second

A car washing service has two parallel washing lanes, each consisting of ***N*** sequential cleaning stations. Each station performs a specific washing or detailing task (e.g., rinsing, foaming, scrubbing, waxing, drying).

A car can start at either lane ***1*** or ***2***, and it must pass through all ***N*** stations before leaving. Each station ***(i, j)*** (where ***i*** is the lane number and ***j*** is the station index) has:

- **Processing time *w[i][j]***: The time required to clean the car at station ***j*** on lane ***i***.
- **Switch time *s[i][j]***: The time required to switch the car from lane **i** to the other lane after station ***j***.
- **Entry times *e[1]* and *e[2]*** denote the time to enter lane ***1*** or lane ***2*** at station ***1***.
- **Exit times *x[1]* and *x[2]*** denote the time required to leave after the last station.

The goal is to determine the minimum time required to wash a car completely.

Input Format

The first line contains an integer ***T (1 ≤ T ≤ 100)***, the number of test cases.

For each test case:

- The first line contains an integer ***N (1 < N ≤ 10^5)***, the number of stations.
- The second line contains two integers ***e1, e2 (1 ≤ e1, e2 ≤ 1000)***, the entry times for lanes ***1*** and ***2***.
- The third line contains ***N*** integers ***w1[1], w1[2], …, w1[N] (1 ≤ w1[j] ≤ 1000)***, the processing times for lane ***1***.
- The fourth line contains ***N*** integers ***w2[1], w2[2], …, w2[N] (1 ≤ w2[j] ≤ 1000)***, the processing times for lane ***2***.
- The fifth line contains ***N-1*** integers ***s1[1], s1[2], …, s1[N-1] (1 ≤ s1[j] ≤ 1000)***, the switching times from lane ***1*** to lane ***2***.
- The sixth line contains ***N-1*** integers ***s2[1], s2[2], …, s2[N-1] (1 ≤ s2[j] ≤ 1000)***, the switching times from lane ***2*** to lane ***1***.
- The seventh line contains two integers ***x1, x2 (1 ≤ x1, x2 ≤ 1000)***, the exit times for lanes ***1*** and ***2***.

Output Format

For each test case, print a single integer – the minimum time required to completely wash a car.

Sample input	**Sample Output**
2 5 2 4 7 9 3 4 8 8 5 6 4 5 2 3 1 3 2 1 2 2 3 2 4 10 12 6 7 8 9 5 6 7 8 4 2 3 2 1 3 3 4	32 42

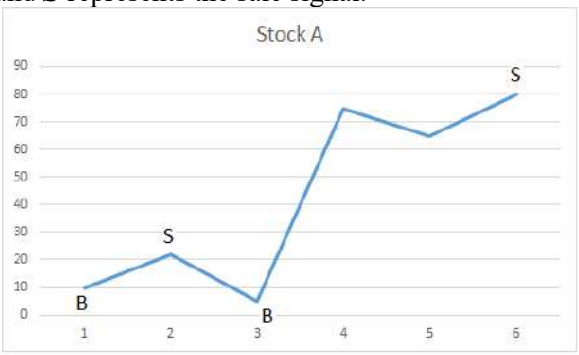
Problem 03: Stock Trader's Dilemma

Time limit: 1 second

These days' stock markets are doing amazing. Investment is very easy, and anyone can become an active trader. Upon looking at such news, CM (Capital Makers) decided to invest in the stock exchange. Having no prior experience, they are in deep trouble in timing the buy and sell signals. They are losing the money of their investors and need your help to come up with a plan to ace the stock market and make profits by buying and selling a stock.

The strategy is to buy a stock, hold it till it gives profit and then sell it. Afterwards, buy the stock once again and sell it. However, since they had already made a lot of mistakes, their shareholders do not trust them anymore. They are restricting them to deal only in one stock at a time and limit the transaction to at most '***k***'. Such that the stock can only be bought '***k***' times and sold '***k***' times. By allowing '***k***' transaction, the shareholders want to have maximum profit.

For example, given the trading data of a stock '***A***' as ***[10, 22, 5, 75, 65, 80]*** and ***k = 2***, the maximum profit can be earned when the stock is bought on day ***1***, sold on day ***2***. Then bought on day ***3*** and sold on day ***6***. Thus, the complete transaction looks like, ***(22-10) + (80-5) = 87***. In the image below, ***B*** represents the buy signal, and ***S*** represents the sale signal.



Input

The first line of the input consists of ***t, (1 ≤ t ≤ 50)*** representing the total number of test cases given. Each test case is represented in three lines. The first line of the test cases represents, ***'k' (0 ≤ k ≤ 10^3)*** allowed trades for the session. The next line contains ***'n' (1 ≤ n ≤ 10^4)*** number of days. The next line contains ***'n'*** values, space separated, representing stock prices on ***'n'*** days.

Output

Output consists of t lines, each line contains exactly one number, representing the maximum profit of the trade.

Sample Input	Sample Output
3 2 6 10 22 5 75 65 80 3 6 10 22 5 75 65 80 1 5 90 80 70 60 50	87 97 0

Problem 04: Calligraphy Crisis

Time limit: 6 seconds

Ahmed is a renowned calligraphy artist who specializes in crafting beautiful Arabic and English letter designs. His shop is famous for custom calligraphy, where customers order personalized artwork with their names or meaningful words.

One day, while Ahmed was busy preparing an order for an important customer, a mischievous kid snuck into his shop and shuffled the letters on his unfinished works! Instead of the elegant, flowing designs, the letters were now in complete disarray.

Ahmed was furious, but there was no time to redraw everything from scratch. Instead, he decided to salvage whatever he could. He needs to quickly check whether any of his existing calligraphy pieces contain a prefix that is an anagram of a customer's requested word.

Can you help Ahmed restore order in his shop by finding which of his artworks can still be used for the incoming customer requests?

Definitions

Prefix: A prefix of a word is any substring of that word that starts at the beginning and can include the whole word itself.

Example: All possible prefixes of "trap" are "t", "tr", "tra", and "trap".

Anagram: Two strings are anagrams of each other if one can be rearranged to form the other.

Example: "abc" and "bca" are anagrams, but "abc" and "bcaa" are not.

Input

The first line contains an integer N — the number of calligraphy pieces Ahmed has in his shop.

The next N lines contain strings S_i representing a calligraphy piece.

The next line contains an integer Q — the number of customer requests.

The next Q lines contain a single word Q_i, representing a customer's request.

Output

For each request Q_i, print the number of calligraphy pieces that contain a prefix that is an anagram of Q_i.

If no such calligraphy piece exists, print "-1".

Limits

$1 \leq N, Q \leq 10^5$

The length of S_i and Q_i is in the inclusive range $[1, 20]$.

S_i and Q_i only include lowercase english alphabets.

The calligraphy pieces are unique, i.e: all N values of S_i are unique.

Sample Input	Sample Output
5 rat art tarp part trap 3 tra ar tp	4 2 -1

Problem 05: BioDiversity Scan

Time limit: 1 second

Global temperature raise is one of the most significant problems in current era. Recently, the world leaders announced Carbon Credits to minimize carbon emissions. One way to monitor Carbon Credits is to check the quality of eco system in a particular region. This quality can be determined by the presence of some specific insects in that region. The most common insects are ladybug, butterfly, and fireflies.

Dr. Wang has developed an AI based automated eco-friendly insects' detection system. This system is installed in a remote area and send the number of total detected insects each day. A minimum of M number of each N insect (ladybug, butterfly, fireflies etc.) is required to maintain a healthy eco system. Let's assume Dr. Wang system detected a total of T insects in one day. Write a program that can determine the number of ways we can have a total count of T for N insects, assuming each insect's category has a minimum of M number. For example, if $N = 3$, $T = 34$ and $M = 10$ then the possible combinations are as follow:

14 10 10
13 11 10
13 10 11
12 11 11
12 10 12
…….
and so on.

Input

In the first line of the input there will be a single positive integer K followed by K lines each containing a single test case. Each test case contains three positive integers denoting $\mathbf{0} \leq N \leq \mathbf{10}$, $\mathbf{0} \leq T \leq \mathbf{200}$ and $\mathbf{0} \leq M \leq \mathbf{10}$ respectively, separated by a single space.

Output

For each input, print the value of total number of possible combinations. Otherwise print 0 if not possible.

Sample:

S.No	Sample Input	Sample Output
1	2 3 34 10 2 40 15	15 11
2	1 30 3 15	0

Problem 06: ICPC Event Management - Optimizing Unique Event Tracking

Time limit: 1 second

ICPC Event Management Company is responsible for organizing various events, including competitions, entertainment shows, parties, processions, dinners, and academic exams. To ensure better accountability and preparation, ICPC needs a robust event tracking system. Each event is categorized by its type and subcategories, allowing for streamlined organization and efficient logistics planning. However, event codes sometimes contain repetitive patterns or **erroneous entries**, making it difficult to track events effectively.

To optimize the system, ICPC wants to determine the longest sequence of unique event codes from a given event schedule. If an **invalid entry** is found, the system should identify the error and flag it for correction.

Additionally, ICPC wants to know the exact **number of instances** of each event category type in the longest sequence to help in better event planning.

Event Categories and Subcategories:

ICPC organizes **7 different types of events**, each with multiple subcategories:

Category	Event Type	Subcategories (Example Codes)
A	Competitions	A01 (Local Contest), A02 (Regional Contest), A03 (National Contest), A04 (World Finals)
B	Entertainment	B01 (Concert), B02 (Movie Night), B03 (Stand-up Comedy), B04 (Theater)
C	Social Gatherings	C01 (Welcome Party), C02 (Networking), C03 (Farewell), C04 (Award Ceremony)
D	Dinners	D01 (Gala Dinner), D02 (Sponsor Dinner), D03 (Corporate Dinner), D04 (Team Dinner)
E	Processions	E01 (Opening Ceremony), E02 (Closing Ceremony), E03 (Cultural Parade), E04 (Community Walk)
F	Training Workshops	F01 (Algorithm Training), F02 (Competitive Coding), F03 (AI & ML Workshop), F04 (Career Guidance)
G	Exams	G01 (Preliminary Round), G02 (Elimination Round), G03 (Final Round), G04 (Certification Exam)

Each event and sub-event are represented by its **category letter** and a **two-digit number** indicating its subcategory. For example, G01 represents **Preliminary Round**, B02 represents **Movie Night**, and D04 represents **Team Dinner**.

ICPC wants to determine the longest contiguous sequence of unique event codes in a given event schedule. If there are multiple longest sequences of the same length, the one that starts with the **smallest event alphabetically** should be chosen.

Additionally, **if an invalid event code is detected**, the program should output -1 followed by the first erroneous entry.

The output should include:

1. **An integer** representing the number of distinct events in the longest unique sequence (or -1 if an error is found).
2. **The list of event codes** in that sequence, space-separated (or the invalid code if an error is found).
3. **The count of each event type category** in the sequence (e.g., 3 Competitions).

Input

- The first line contains the number of test cases.
- For each test case a single string *S*, where $0 < |S| < 1001$, with $|S|$ representing the total number of events.

Output

- Each line corresponds to the output of one test case, in the same order as the input. i.e.
    - If all event codes are valid:

- An integer representing the number of distinct events in the longest unique event sequence.
- The longest unique sequence of event codes.
- The exact number of instances per event type in the sequence.

- If an invalid event code is found:
  - -1 followed by the first incorrect entry of length 3 where possible.

Sample Input	Sample Output
5 A01B02C03D04 A01A02A03 G01G02G03G04 A01A01A02A01 A01A02A3D02F09	4 A01 B02 C03 D04 1 Competitions 1 Entertainment 1 Social Gatherings 1 Dinners 3 A01 A02 A03 3 Competitions 4 G01 G02 G03 G04 4 Exams 2 A01 A02 2 Competitions -1 A3D

Problem 07: Electoral Boundaries

Time limit: 1 second

The changing dynamic of All Community Housing Society forces an early re-election for the president position. The society is very large and consists of many different blocks uniquely identified by a number ***1*** to ***N***. Some blocks are connected with each other and some not making them isolated. The election commission of the society received various suggestion for setting up polling booths across the society. The society wants to facilitate all the residents of the blocks to cast their votes at their ease.

The election commission wants to divide all these blocks into ***M*** distinct polling districts. However, there's a special requirement for the way these districts are organized:

- Every block must be assigned to exactly one polling district.
- For any two blocks that are directly connected by a road, the polling districts they belong to must be consecutive. In other words, if one block is placed in district ***X*** and its connected neighbor is in district ***Y***, then the absolute difference between ***X*** and ***Y*** must be exactly ***1***.

The challenge for the election commission is to determine the maximum number of polling districts ***(M)*** that can be created to facilitate voters. If it turns out to be impossible to arrange the blocks into such districts, the answer should be ***-1***.

Input

The first line of the input consists of ***t, ($1 \leq t \leq 50$)*** representing the total number of test cases given. The first line of the test cases represents, ***'N' ($2 \leq N \leq 10^3$)*** representing ***'N'*** blocks. The next line contains ***W ($1 \leq W \leq 10^6$)*** representing number of connecting roads. The next W lines contains two space separated block numbers representing a road between the two blocks.

Output

Output consists of t lines, each line contains exactly one number, representing the maximum number of distinct polling districts ***(M)***.

Sample input	Sample Output
4 6 6 1 2 1 4 1 5 2 6 2 3 4 6 3 3 1 2 2 3 3 1 3 1 2 3 6 5 1 2 2 4 1 5 4 3 3 6	4 -1 3 6

Problem 08: Optimized Path Country

Time limit: 1 seconds

The country of Zelda has an extensive motorway system and has implemented a smart traffic system on its network. As soon as a driver enters a road network, an automated system demands the driver's destination to work out the exit toll gate and compute the optimal path. The automated system scans the car's tag and generates a coded sequence giving the optimal path and asks the driver to set his/her mileage to zero. In addition, the receipt also contains a token number which will be useful at the exit gate.

At the exit gate, the smart city has implemented a clever solution. If the driver has followed the optimal path, the car's tag will contain the total optimal distance covered as well as a number generated from the token that the car received at entrance. This is computed by adding the highest three prime numbers that are lower than the token. The toll is subsidized if the driver followed the optimal path, and a penalty is added if he/she took a longer route.

You want to take a romantic long drive on the motorway network but still pay a reduced toll. To do that, you need to enter the distance of the optimal path the code number. You are not to use any loops in your travel. Your point of entry is always denoted as town 1.

Input

In the first line has the road towns/intersection $N\ (1 \le N \le 100)$, followed by a number R corresponding to the number of roads in the network $R\ (1 \le R \le N \times N)$, your desired exit point $E\ (1 \le E \le 100)$, and the token assigned to you $T\ (1 \le T \le 32000)$. The subsequent R lines contain the distances $d\ (1 \le d \le 1000)$. corresponding to the road distances from towns u to $v\ (1 \le u,v \le 100)$.

Output

Your output should have as many lines as the number of test cases. Each line indicates the distance of your chosen path followed by the generated value from the token. In case there is no such path to your destination, the distance should be 2147483647.

Sample Input	Sample Output
4 4 3 10 1 2 2 1 3 3 2 3 4 3 4 1	3 15
4 2 3 15 1 2 2 3 4 1	2147483647 31

Problem 09: Ancestral Queries

Time limit: 1 second

The Khan family submitted their digital family tree to the Lahore Heritage Society. The judges were impressed by how efficiently the family had documented their ancestry using algorithmic thinking. They even asked Zara to present her solution at the Lahore Tech Expo, where it became a hit among historians and tech enthusiasts alike.

The Khan family tree looked like this:

- **Dada Abu**: The patriarch of the family.
  - His children: **Ali** and **Fatima**.
    - Ali's children: **Hassan** and **Ayesha**.
      - Hassan's children: **Bilal** and **Sana**.
        - Bilal's children: **Amna** and **Usman**.
    - Fatima's children: **Zainab** and **Omar**.
      - Zainab's children: **Leena** and **Yusuf**.
      - Omar's children: **Hania** and **Saad**.

With the digital family tree in place, the Khan family organized a **grand reunion** at their ancestral home in Lahore's **Old City**. Family members from all over Pakistan gathered to celebrate their shared history and learn more about their ancestors.

- **Amna** discovered that her 4th ancestor was **Dada Abu**, the patriarch of the family.
- **Leena** learned that her 2nd ancestor was **Fatima**, her great-grandmother.
- **Saad** was amazed to find out that his 3rd ancestor was **Ali**, a prominent businessman in Lahore's history.

The Khan family's story became a symbol of **unity, heritage, and innovation** in Lahore. Their digital family tree not only helped them connect with their past but also inspired other families in the city to document their own histories using technology. As Dada Abu often said, **"A family's roots are like the foundations of a building. The stronger they are, the taller we can rise."**

The story highlights the importance of family, heritage, and technology in preserving our connections to the past. Just like the binary lifting technique helps us efficiently find ancestors, understanding our roots helps us navigate the present and future with clarity and purpose.

Description:

Given a family tree of ***n*** members and ***q*** queries, for each query, find the ***k-th*** ancestor of a given family member. If the ***k-th*** ancestor does not exist, return ***-1***.

Input:

- The first line contains the number of testcases (***t***).
- The second line contains ***n*** (number of family members), ***q*** (number of queries) and ***r*** (root of tree).
- The next ***n−1*** lines describe the parent-child relationships in the family tree.
- The next ***q*** lines contain pairs **(*u*, *k*)**, where ***u*** is the family member and ***k*** is the ancestor level.
- All numbers in a line are single space separated.

Output:

- For each query, print the ***k-th*** ancestor of ***u***. If it doesn't exist, print **-1**.

Input	Output
1 5 2 5 5 3 1 3 4 3 1 2 4 2 1 3	5 -1