The 2023 ICPC Asia Topi Regional Onsite 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 questions regarding the problems, send 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.**
- Unless mentioned in some problem, all your programs must meet the time constraint of 5 seconds.
- The decision of judges will be final.

Problem 1: Smart Pricing

Time limit: 3 seconds

Australia is known for its love for sports and healthy lifestyle. The young Australians are very cautious about their diet and choice of food. Normally, in an Australian supermarket one can easily find several choices of the same food item, according to the nutrition and ingredient mix. In addition, there are many well established family businesses with a unique taste and association of customers.

Mr. Jamal is a young and enthusiastic entrepreneur who is running a 40 years old family owned fresh fruits juice and beverage business in Melbourne. They have a unique taste and a large base of customers all over Australia. This business was established by Jamal's grandfather, who migrated from Lebanon back in 1940. Jamal's grandfather uses a special secret ingredient to create a unique taste in beverages. He named that secret ingredient as "Lebon", which passed from generation to generation.

With the advancement of new technologies, Mr. Jamal wanted to set the price of his product (juices) according to the packing size and the amount of secret ingredient "Lebon" used in them. Interestingly, the small size packing bottles cost more than large size bottles due to the amount of labor required to prepare and pack them. In addition, other charges also add up to decide the end product price.

For this purpose, Mr. Jamal labeled the prices by assigning some parameters to each product. Given a list of products, each represented by a tuple containing the product name, bottle size (in liters), amount of Lebon used (in grams), the sticker cost, and the amount of fruit used (in kilogram KG). Interestingly, each product has a different amount of fruit used. As a good friend, Mr. Jamal asked you to find the longest possible sequence of products in the data with increasing bottling size and decreasing Lebon. The product with the lowest bottling-size to Lebon ratio should be considered the "cheapest". If there is a tie, the product with the lowest sticker cost should be considered as "cheapest". If there is still a tie, the product that uses the smallest amount of fruits should be considered "cheapest". Note that it is possible to skip a product when generating the sequence.

Input

The first line of input is an integer ***N (4 <= N <= 100)***, representing the total number of product entries a user wants to enter. The next ***N*** lines represent the input data by the user. The input data for each product is a ***5-tuple*** of the form (name, bottling-size, Lebon, sticker-cost, fruits), where each value is separated by a space. The list represents a single product's data per line. Each product is labeled with a different name. All the input values are integers.

Output

The first line should print an integer ***K***, indicating the length of the longest found sequence. The next ***K*** lines should print the name of the product in order of cheapest price. Such that the cheapest found product is on the top followed by the second cheapest product in the sequence and so on. Let ***j*** represent the output sequence order, such that ***j*** = 1 represent the cheapest product in the sequence, ***j*** = 2 is the second cheapest and so on, the output sequence must follow

$$B(1) < B(2) < B(3) \dots\dots < B(K) \qquad \text{and} \qquad P(1) > P(2) > P(3) \dots\dots > P(K)$$

Where ***B*** is the bottling-size and ***P*** represent the price of the product. For your answer to be correct, ***K*** must be as large as possible. It's guaranteed that there will only be one possible solution with max ***K***.

Sample input & output

The following is an example of a sample input and corresponding correct outputs.

Sample input	Sample Output
Sample 1:	
6 P 7 130 5 3 M 6 210 3 2 L 5 200 7 5 E 1 400 4 7 D 2 300 3 6 J 3 190 8 5	4 E D J P
Sample 2:	
4 C 4 130 7 2 J 2 200 9 4 W 1 180 7 5 F 2 160 5 5	3 W F C

Problem 2: Builder's Dilemma

Time limit: 5 seconds

Maverick Díaz is a well known real-estate developer in the city of X. His great skill at cutting costs and efficient use of resources to build projects has gathered him unreal amounts of wealth. Mr. Díaz owns multiple buildings scattered throughout the city. Each building has some height ***H*** and some cost ***C***. Mr. Díaz recently got into an argument with his chief architect and ended up firing him. Now, Mr. Díaz thinks that he should be able to re-use architectural plans for the existing buildings to come up with new building plans.

A new building plan can be created by combining two existing building plans, given that the plans are compatible with each other. Each building plan has some height ***H*** and some cost ***C***. Let's say we have two mutually compatible building plans ***P*** and ***Q***, having heights H_P and H_Q respectively and having costs C_P and C_Q respectively. If we merge the building plans **P** and ***Q*** to create a new plan ***R***, then the height H_R of the new building plan is the sum of the heights of the plans merged, and the cost C_R of the new plan, is the product of the cost of the plans merged. To put it simply: $H_R = H_P + H_Q$ and $C_R = C_P \times C_Q$

Mr. Díaz regularly constructs new buildings. Whenever he wants to construct a new building, he first decides the required height **H** for the new building. In his line of work, it is cheaper and much more efficient to construct multiple buildings of the same height together. So, he always constructs all the different buildings meeting the height requirement by combining existing building plans.

You are an analyst working for Mr. Díaz. You have access to two lists L_1 and L_2 of building plans. Each plan from L_1 is compatible with each plan from L_2, but any plan from one list is not compatible with any other plan from the same list. Mr. Díaz wants you to answer his queries about the cost of new buildings. Answer those queries otherwise you might be next in line to get fired. Note that you can re-use plans between queries.

The problem statement sounds easy but solving it with a time-efficient algorithm for big values of ***N*** and ***M*** will be difficult and brute-force solutions will not be accepted.

Input

First line contains two integers ***N*** and ***M***, denoting the length of lists L_1 and L_2.
Second line contains ***N*** integers, where each integer C_i denotes the cost of the i^{th} plan of L_1.
Third line contains ***N*** integers, where each integer H_i denotes the height of the i^{th} plan of L_1. All height values are unique in the list.
Fourth line contains ***M*** integers, where each integer C_i denotes the cost of the i^{th} plan of L_2.
Fifth line contains ***M*** integers, where each integer H_i denotes the height of the i^{th} plan of L_2. All height values are unique in the list.
Sixth line contains a single integer ***Q***, denoting the number of queries.
Seventh line contains ***Q*** integers, where each integer Q_i denotes a height query.

Limits

$1 \leq N, M \leq 105$
$0 \leq H \leq 105$
$0 \leq C \leq 32$
$0 \leq Q, Qi \leq 2 \times 105$
All height values are unique within their respective lists

Output

For each query Q_i, print the total cost to construct all possible buildings having height Q_i. If such a building is impossible (i.e: no two plan heights add up to Q_i), then print 0.

Sample input	Sample Output
2 3 2 3 1 2 6 4 5 2 0 1 3 0 3 4	0 27 18

Sample Case Explanation

In Query 1, there's no way to construct a building having total height 0, hence the cost is 0.

In Query 2, we want to construct a building having height 3. We can construct two buildings having this height. The first way is to merge plans for building having height 1 in L_1 and the building having height 2 in L_2. The cost for this way is 2 x 6 = **12**. The second way is to merge the plan having height 2 in L_1 and the plan having height 1 in L_2. The cost for this way is 3 x 5 = **15**. The total cost is then: 15 + 12 = **27**.

In Query 3, we want to construct a building having height 4. There's only one way to construct such a building. We merge the plans for building having height 2 in L_1 and the building having height 2 in L_2.

Problem 3: Inverted Reality

Time limit: 2 seconds

You find yourself in an alien world where everything is the inverse of Earth. Left is right, right is wrong, short is tall, black is white. Fish soar through the sky, elephants scale trees, birds bark, and dog's chirp. The aliens, aware of Earth's advanced technology, offer you a chance to help them develop similar technologies. Your first task is to design a machine capable of basic arithmetic and logic operations. They provide you with an Earth laptop for this task. However, remember that in this world, addition means subtraction, multiplication means division, and big is small.
Your mission is to create an algorithm that can perform simple arithmetic and logic operations. You need to write code that performs a series of operations on a list of numbers, but in reverse. For example, given the input [1 2 1 -1] and asked to add 2 and multiply by -2, the output should be [0.50 0.00 0.50 1.50].

Input

The first line of input is an integer n ($1 \leq n \leq 100$), the number of test cases. The second line is the first test case T_1, a list of k real numbers where $-100 \leq T_{1i} \leq 100$, for $1 \leq i \leq k$. This is followed by an integer, op, the number of operations to be performed on T_1. The following op lines contain a real number (operand) and a character (operator). Operators are only be chosen from set i.e. $\{+, -, *, /, <, >\}$. The sequence of operation is <Input> <Operator> <Operand>.

Output

For each test case i, print the output after all the arithmetic/logic operations given for the test case. The output should be a list of real numbers (floats), rounded to two decimal places. For example, 1.00 2.33 0.00.

Sample Input	Sample Output
2 1 2 1 -1 2 2 + -2 * 2 5 3 -2 20 6 3 < 5 - 2 * -2 + 3 - 5 /	0.50 0.00 0.50 1.50 37.50 40.00 37.50 37.50 40.00

Problem 4: Spy Message Transformer

Time limit: 2 seconds

A security agency has its spies spread across the country for surveillance purposes. They require a mobile app that will be used to send messages to the command center. This app will transform messages into an unreadable form using their internal algorithm.

The message can contain alphabets ***[a-z]***, numbers ***[0-9]*** and special characters. The characters in every word should be reversed, and then the order of words in the message should be reversed. To hide the spaces in the message, replace them with the length of the smallest word. Finally, your program should print the transformed message.

Input:

The first line contains the number of test cases ***0<N<100***. Each of the subsequent N lines is a string representing a meaningful message which may contains alphabets ***[a-z]***, numbers ***[0-9]*** and special characters.

Output:

Output is string message in non-readable form which contains alphabets ***[a-z]***, numbers ***[0-9]*** and special characters. It should not contain any spaces.

Sample Input	**Sample Output**
3 Where are we going? I can't reach there I am work and waiting for your response!	?gniog2ew2era2erehW ereht1hcaer1t'nac1I !esnopser1ruoy1rof1gnitiaw1dna1krow1ma1I

Problem 5: Bandit Country

Time limit: 1 seconds

You must cross a dangerous country full of bandits. A map is provided to you at the border using which you must plan your route. You are low on fuel and would like to take the shortest path. However, there are bandits with insiders at the point of entry who inform their accomplices of your planned exit point from the country. The country's cities are relatively safe, but the highways are notoriously dangerous. Knowing your entry and exit points, the bandits make a head start and set traps on your potential route where they may try to target you. With insider information on your fuel levels, they know that you may take the shortest route and target those highways. There is only one direct highway between any two cities. Hence, you must try to avoid those highways while finding an alternate shortest route.

If you're not able to find an alternate route, you will need police escort which is very expensive to hire, and you would rather avoid it if you can find an alternate route. Your solution should provide the total length of your route and the cities you'll be touching or, in case there is no such path, indicate that police escort will be needed. The following figure corresponds to the input/output given below.

(a) (b)

Figure 1: Map corresponding to a country with four (04) cities which are represented as nodes. The edges correspond to the length of the highway between any two cities.

Input

The first line in the input file is the number of test cases, ***N*** ***(1 ≤ N ≤ 100)***. Each of the *N* subsequent lines represent a test case. Each subsequent line has a few entries. The first integer is a number ***M*** representing the number of cities in the country, the second integer is the start point, ***S***, and the third integer is the exit point, ***E*** with ***1 ≤ S, E ≤ 100***. This is followed by the list of ***M × M*** entries representing the length of the highway between cities. The values are read in a row-wise fashion and a value of 0 indicates that no highway exists. All input entries are space separated.

Output

Your output should have as many lines as the number of test cases. Each line indicates the length of your chosen path followed by the path, i.e., sequence of cities from the entry to the exit point. All values in a line should be space separated. In case no alternate path is found, it should output "***-1***".

There are two test cases corresponding to figure 1(a) and 1(b), respectively. Both test cases have 4 cities, you enter at city #1 and exit at city #4. In the first test case, the length of your route is 3 and the path chosen is city #1 to city #2 and then to city#4. For the second case, no such route is possible.

Sample Input	Sample Output
2 4 1 4 0 1 1 0 1 0 3 2 1 3 0 2 0 2 2 0 4 1 4 0 1 0 0 1 0 3 2 0 3 0 2 0 2 2 0	3 1 2 4 -1

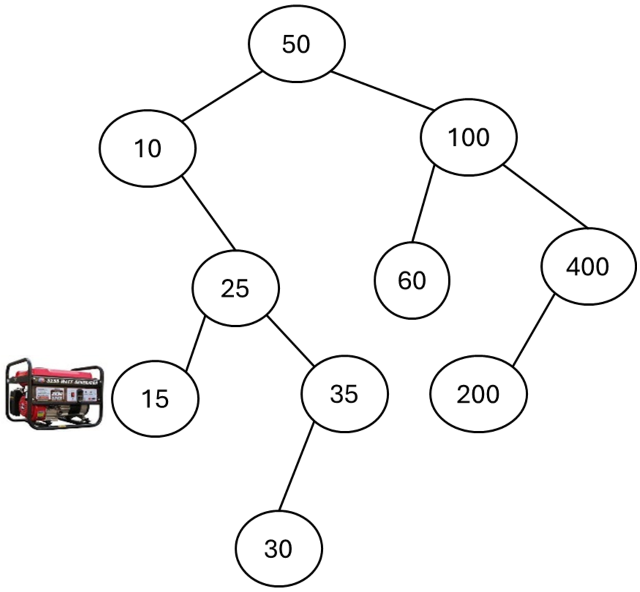
Note:

- *In case of equal distances when calculating paths, use the path that uses the lesser number of cities.*
- *In case of the same number of cities, go to the next higher index (e.g., in the case you can go from 1 to 2 or 3, choose 3).*

Problem 6: Lighting Up City by Cheema & Joyia Power Plant

Time limit: 3 seconds

Cheema & Joyia Power plant is interested in installing an electricity generator in Faisalabad city. The electricity network in the city follows a binary search tree structure. The company must decide where to place the generator so that the lighting of different points in the city starts in a certain pattern at nighttime. The scenarios are explained in the figure given.



Input

There are ***N*** points where city light points where **0<*N*<100**. The light points are input as Binary Search Tree where each light input number is unique. In the example given, the first line will show how many city points are available in the city. The next line shows the installation site where the generator is installed. The subsequent line shows the ***N*** light points separated with a single space.

Output

The output will display the sequence of city point numbers indicating the order in which lights will be turned on. The first line will show the installation site, and subsequent lines will indicate the next points that will be illuminated. For example, if the installation site is 15, the first point to light up will be point 15, followed by point 25. Since two points are connected to point 25, points 35 and 10 will illuminate simultaneously.

Note:

- If the node being illuminated has both a parent and a child node to be illuminated next, the name of the child node will be displayed first, followed by the parent node.
- If the node being illuminated has two child nodes, the name of the left child node will be displayed first, followed by the right child node.

Sample input	**Sample Output**
10 15 50 10 100 400 25 35 30 15 60 200	15 25 35 10 30 50 100 60 400 200

Problem 7: Strategic Problem Solving

Time limit: 2 seconds

In the International Collegiate Programming Championship (ICPC), you and your team face a series of problems, each with its own score and estimated time to solve. Your objective is to maximize your team's total score within the contest's limited timeframe. However, each problem can only be solved once, and the time spent on each problem is crucial. Your challenge is to select the most advantageous set of problems to solve, optimizing your score while adhering to the time constraints.

Input

The first line of the input contains two integers N and T, where N $(1 \leq N \leq 100)$ is the number of available problems and T $(1 \leq T \leq 300)$ is the total time available for the contest in minutes.

Each of the next N lines provides details about a problem, containing two integers: the problem's score S $(1 \leq S \leq 1000)$ and the estimated time t $(1 \leq t \leq 200)$ in minutes required to solve it.

Output

Produce a single integer, the maximum total score that can be achieved within the given time limit.

Sample input	Sample Output
4 300 500 150 800 200 400 100 300 120	1200

Problem 8: Raja Bazar

Time limit: 8 seconds

Raja Bazar in Rawalpindi boasts a labyrinthine road network that intricately weaves through the heart of the city. Navigating this bustling urban landscape requires a keen understanding of its complex web of streets, alleys, and intersections.

You are a traffic engineer working for a city's transportation department, and you're tasked with analyzing road network patterns in a complex road network. Your goal is to identify and count specific road network structures, within the road network.

A Road Network Structure is a recurring subgraph pattern that represents a specific topology of road network. Understanding these road network structures can help optimize traffic signal timings and reduce congestion.

In this problem, you are given an undirected road network consisting of road segments, and intersections. Where intersections are represented by nodes and road segments are represented by edges. Your task is to discover and count all occurrences of a particular Road Network Structure within the given road network. Multiple Road Network Structures may share road segment(s) and/or intersection(s). Two Road Network Structures are distinct when they contain at least one distinct intersection.

Input

- First line contains a pair ***N***, ***M***. The number of intersections in the road network, ***N*** **(**$1 \leq N \leq 200$**)**. The number of road segments in the network, ***M*** **(**$1 \leq M \leq (N(N-1)/2)$**)**.
- ***M*** lines, each containing two intersection IDs ***u*** and ***v*** **(**$0 \leq u, v \leq N-1$**)**, representing an undirected road segment from intersection *u* to intersection *v* in the network and vice versa.
- Pair of integers ***K***, ***P***. The number of intersections in the target road structure ***(***$1 \leq K \leq 5$***)*** followed by ***(***$1 \leq P \leq (K(K-1)/2)$***)*** the number of road segments involved in the target road structure.
- ***P*** lines, each specifying a pair of intersection IDs ***u*** and ***v*** ***(***$0 \leq u, v \leq N-1$***)***, representing road segments from intersection ***u*** to intersection ***v*** and vice versa within the road network.

Output:

Output a single integer, the count of occurrences of the target structure within the provided road network.

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 3 0 1 1 2 2 0	3

Problem 9: Agri Yield Optimization

Time limit: 3 seconds

Imagine you are working on a project to optimize agricultural yield in a huge landscape. The landscape is used for different crops by a company in smart agricultural domain. The growth rate of each crop is represented in a static array of a fixed size, 8 i.e. ***[2,4,1,5,3,6,7,8]***. Each element in the array represents the growth rate of a specific crop at a given position and each position in the array represents the fields in the landscape. However, due to the tighter financial situation, the company wants to use the sets of contiguous fields [range of indexes, in the form of two indices; left and right] to cut down expenses.
Your task is to design an algorithm using the product of the square root to calculate the growth yield of contiguous fields. The growth yields will help the company to identify optimal contiguous regions for planting crops. One simple approach to solve this problem is to use brute force to find the overall growth yield within the given set of contiguous indexes in the array presented as a query.
However, to calculate the overall growth yield optimally in the given range you can decompose the array into components in the following way:

Number of Components = ***floor ((8 / component_size))*** where ***component_size =floor(sqrt(8))***

For keeping the accuracy, you are required to use 6 decimal places of precision for all calculations. Further, the final answer shall be represented in 3 decimal places of precision. You are also required to count the number of component hits required to reach the final answer.

Input

The first line of the input file is the number of queries ***N (1 ≤ N ≤100)***. Each subsequent line, represents the user query (values of left and right index) for your solution to calculate the potential growth of crops.

Output:

For each query, two values should be output, where first represents the potential growth and second represents the number of component hits required.

Sample input	Sample Output
3 1 6 0 7 1 5	50.2 4 200.798 4 18.974 3