Problem Set

The 2022 ICPC Asia Topi Online Preliminary Programming Contest

Instructions

- Do not open the booklet unless you are explicitly told to do so. You can only read these instructions below.
- Do not create disturbance or move around unnecessarily in the arena.
- If you have any question regarding the problems, send a clarification from the judges using DOMJudge.
- There would be no internet access and mobile phones are also not allowed.
- 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 absolutely final.

Problem 1: Hello Quetta

Time limit: 5 seconds

Test program to print the Hello Quetta.

Input:

First line contains the number of sample strings, ***N***. Each of the ***N*** subsequent lines after that represent the string to be printed.

Output:

Each input string should be output as “Hello %s!\n”

Sample Input	**Sample Output**
3 Quetta ICPC Ignite	Hello Quetta! Hello ICPC! Hello Ignite!

Problem 2: Movement of Knight

Time limit: 5 seconds

King of Kharasan is very fond of chess. He used to play aggressive moves with knight. In his plans, he often wants to move knight on certain position in minimum moves. In chess, knight always moves either two horizontal steps followed by one vertical step or two vertical steps followed by one horizontal step in any direction within the board. For example, if knight is at cell (3, 3), it can visit any one of (4, 5), (5,4), (4,1), (5,1), (2,1), (2,5), (1,2), or (1,4) location.

Chessboard has 8x8 blocks/ cells. King is interested to find out the minimum number of moves knight can take to move from a source block/ cell to a destination block/ cell, where some blocks/ cells are already occupied by other pieces.

Input:

The input consists of multiple test cases. The first line is the number of test cases $\boldsymbol{T}$ ($1 \leq \boldsymbol{T} \leq 5000$). After that, there exist $\boldsymbol{T}$ test cases. Each test case has three lines as follows:

- First line has source position of knight r ($0 \leq r \leq 7$), c ($0 \leq c \leq 7$) followed by target position of knight r ($0 \leq r \leq 7$), c ($0 \leq c \leq 7$).
- Second line has n ($0 \leq n \leq 60$) number of occupied blocks.
- Third line has n pairs of r ($0 \leq r \leq 7$), c ($0 \leq c \leq 7$) that is position of occupied blocks.

Output:

For each test case, output Case #: followed by number of shortest moves to reach from source to destination block.

Sample Input	**Sample Output**

3
1 1 5 5
2
2 3 3 2
1 1 5 5
6
2 3 3 2 5 1 3 4 3 6 4 3
1 1 7 7
2
2 3 3 2
	Case 1: 4
	Case 2: 6
	Case 3: 6

Problem 3: PSL Final Match

Time limit: 5 seconds

Bilal Khan and Mansoor Achakzai are very excited for Pakistan Super League (PSL) final cricket match in National Cricket Stadium, Karachi. Due to limited capacity in the stadium, the tickets have been sold out and they couldn't get the tickets for the final match between Quetta Gladiators and Karachi Kings. They decided not to give up easily and plan to watch the match from their house's top floor from which the stadium ground is visible.

The top floor is surrounded by a safety wall with a height $\boldsymbol{Y}$ and there are $\boldsymbol{N}$ blocks of cement (numbered $\boldsymbol{1}$ through $\boldsymbol{N}$) of different heights such that for each valid $j \in [1, \boldsymbol{N}]$, the j^{th} cement block has a height, $\boldsymbol{H_j}$.

Bilal wants to use some cement blocks and stack them on top of each other to build two towers. The height of each tower is the sum of heights of all the blocks that form it. Of course, no block may be in both towers. The height of each tower should be at least $\boldsymbol{Y}$. Then, Bilal can climb on top of one tower and Mansoor on top of the other and they can watch the match uninterrupted.

While Bilal is busy in bringing the blocks, Mansoor would like to know the smallest number of blocks required to build two towers such that each of them has height at least $\boldsymbol{Y}$ to build such towers. Can you help Mansoor?

Input

The first line of the input contains an integer $\boldsymbol{T}$ $(1 \leq \boldsymbol{T} \leq 5)$, which denotes the number of test cases. Then the $\boldsymbol{T}$ test cases follow with the following description (each test case consists of two lines):

- The first line of each test case contains two space-separated integers $\boldsymbol{N}$ $(1 \leq \boldsymbol{N})$ and $\boldsymbol{Y}$ $(\boldsymbol{Y} \leq 3{,}000)$.
- The second line contains $\boldsymbol{N}$ space-separated integers $\boldsymbol{H}_1, \boldsymbol{H}_2, \ldots.., \boldsymbol{H}_N$. $(1 \leq \boldsymbol{H}_j \leq 105$ for valid $j)$

Output

For each test case, print a single line containing one integer - the minimum number of blocks required to build both the towers, or **−1** if it is impossible.

Sample Input	Sample Output

3
10 40
5 17 4 18 9 8 7 6 20 10
5 16
2 4 10 9 18
4 56
2 4 10 3
	7
	3
	-1

Problem 4: LazyCashier

Time limit: 2 seconds

LazyCashier works at a supermarket where his job is to supervise all the other cashiers. He soon realizes that the cashiers at his store are not good in arithmetic and prone to making mistakes while return change after their customers have paid. He decides to remove the label of being called lazy and offered to do all the calculations. Each cashier would tell him the bill the customer had to pay, and the actual amount paid by the customer. LazyCashier would then tell them the exact change to be returned along with the denominations of the cashback using the minimum number of bills to give the change. It's now your job to write a code for the LazyCashier to be able to compute these answers.

The rupee denominations as of 2022 are 5000, 1000, 500, 200, 100, 50, 20, and 10, 5, 2, and 1.

Input

The first line of the input is ***N***, the number of test cases. Each subsequent line corresponds to a test case and consists of the actual bill of the customer and the amount given by the customer. 1000). The maximum bill and payment can each be Rs. 1,000,000. Both bills and payments are to the nearest whole number only.

Output

Each line will correspond to the output of the test cases (hence ***N*** lines). An output would give the change to be returned and the denominations of Rs. and the number of such currency notes to be returned (0 if no such note is to be returned) using the forma given in the sample output (where "change" is the amount to be returned, "space" indicates there is a space, "Y" and "Z" are the number of denominations):

change: space (5000xY) space + space (1000xZ) …..

Sample input	**Sample Output**

2
3270 5000
820 1000
	1730: (5000x0) + (1000x1) + (500x1) + (100x2) + (50x0) + (20x1) + (10x1) + (5x0) + (2x0) + (1x0)
	180: (5000x0) + (1000x0) + (500x0) + (100x1) + (50x1) + (20x1) + (10x1) + (5x0) + (2x0) + (1x0)

Problem 5: Group Division for Tug of War

Time limit: 2 seconds

A group of students decide to play a tug of war. The students would need to be divided into two teams. Each student must be on one team or the other; the number of students on the two teams must not differ by more than one; the total weight of the students on each team should be as nearly equal as possible.

Input

The first line of the input is ***N***, the number of test cases. Each test case has two lines, as mentioned below:

- The first line is the number of students (***S***)
- The second line of the test case are the weights of the ***S*** students, separated by a space.

Output

For each test case, each output will be a single line containing two numbers: the total weight of the people on one team, and the total weight of the people on the other team. If these numbers differ, print the minimum first. The following table shows a sample input and output format.

Sample input	**Sample Output**

2
5
80 60 120 50 60
4
15 20 90 100
	180 190
	110 115

Problem 6: Rod Cutting

Time limit: 5 seconds

Given a rod of length ***M*** inches and an array of prices that includes prices of all pieces of size smaller than ***M***. Determine the maximum value obtainable by cutting up the rod and selling the pieces. For example, if the length of the rod is 8 and the values of different pieces are given as the following, then the maximum obtainable value is 22 (by cutting in two pieces of lengths 2 and 6).

length	1	2	3	4	5	6	7	8
price	1	5	8	9	10	17	17	20

And if the prices are as following, then the maximum obtainable value is 24 (by cutting in eight pieces of length 1).

length	1	2	3	4	5	6	7	8
price	3	5	8	9	10	17	17	20

Input

The first line of the input is ***N***, the number of test cases. Each test case has a single line corresponding to the price. Each element of the row corresponds to increasing length starting from 1, i.e., first entry is the price of length = 1, second entry is the price of length = 2, and so on. The entry for each case is done such that the first element is ***M*** which is followed by the ***M*** entries, separated by a space as shown below. Assume the maximum length cannot be more than 500. The sample input below corresponds to the entries for the two examples given above.

Output

Each line will correspond to the output of the test cases (hence ***N*** lines). The output is the maximum price you can get selling the rod.

Sample input	**Sample Output**

2
8 1 5 8 9 10 17 17 20
8 3 5 8 9 10 17 17 20
	22
	24

PROBLEM 7: Happy Customers

Time limit: 5 seconds

In wake of high demand of dates in the holy month of Ramadan, the Sabeel Cash and Carry introduced a new packaging scheme for customers on discounted prices. The dates will be available in two different packages of size ***A*** and size ***B***. The sizes ***A*** and ***B*** are decided by staff as per the demand. However, there are limited stock available for each day.

Given the size of the packages ***A*** and ***B***, the total quantity of dates stock ***D*** and the number of customers ***N***, find the maximum number of customers that can be served with the given dates stock in a single day.

You should display the maximum number of customers that can be served along with their index. Assume the index of customers starts from 1.

Input

The first line of input contains two integers ***N*** *(no. of customers)* and ***D*** *(total stock available)*; next line contains two integers denoting the sizes of packages ***A*** and ***B***, *respectively*. The next ***N*** lines contain two integers for each customer denoting total number of packages of size ***A*** and size ***B*** (separated by a space) that customer requires; where ***N <= 100***.

Output

Print maximum number of customers that can be served and in next line print the space separated indexes of served customers in serving order. Please note that the customer with minimum servable order will be printed first, followed by the second minimum servable order, and so on. In case of similar sized order, the customer with the lower index will be printed first.

Sample input & output

Sample input	**Sample Output**

6 4
1 1
2 0
3 2
4 4
10 0
0 1
4 2
	2
	5 1

PROBLEM 8: Faulty AutoWaiter Drone

Time limit: 3 seconds

The “*Aerial Robotics and Vision Lab*” at GIK Institute has developed an in-house drone “*Automatic Waitedrone*” that can perform all duties of a waiter including delivering food and liquids and picking up utensils from the table. The lab has also developed a customized version of the robot called “*Circular AutoWaitor*” that can automatically determine the number of participants sitting in a circle and serve them.

The lab has donated a *Circular AutoWaitor* to ICPC Pakistan Chapter management, who are using this aerial drone to serve participants in the annual dinner. The participants are seated on a table in a circular manner. However, due to a technical fault, the drone does not pick all the utensils in a sequential order after the dinner. Instead, it will pick up utensils from the first person, skip one person, pick up utensils from the third person, skip the next two persons, and so on until all seats are cleared. Once the utensils from a seat have been picked, the drone does not include the cleared seat in the skip count. Since the participants are sitting in a circle, the process of picking up utensils stops when all the seats are cleared.

For example, if 5 participants are seated, then the utensils will be picked up in the following order – 1 3 2 5 4, and the utensils are taken last from seat number 4.

Similarly, if 4 participants are seated, then the utensils will be picked up in the following order – 1 3 4 2, and the utensils are taken last from seat number 2.

Given the number of people sitting at the table, can you help ICPC identify the seat from which the Circular AutoWaiter will pick up the utensils last?

Input

The first line gives the number of test cases n (0 < n < 1000), which is followed by n lines for each test case. Each test case contains a single entry – the total number of people p (0 < p < 5000) sitting in the round table.

Output

The output of the program contains n (0 < n < 1000) lines, one for each test case, indicating the seat from which the utensils will be picked up last.

Sample input & output

Following is a sample input and corresponding correct output.

Sample input	Sample Output
5	4
5	4
7	8
12	11
15	13
25	