Problem Set

ICPC Programming Competition 2020-21

Day 1

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 PC^2.
- 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++". In particular, 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 Python 2 & 3
- Source code file name should not contain white space or special characters.
- You will have to make an Input file for every problem based on the format provided in the Sample Input. For example, the input file for Problem 1 shall be "P1_input.txt"; for problem 2, it shall be "P2_input.txt"; and so on. The outputs of all the problems would have to be on the console according to the sample output given in
- While programming, do not open input files by providing absolute/complete paths (absolute paths of your machine and judge's machine would be different). Just open them from the current directory from where your code is running.
- Please, don't create/open any file for output. All outputs must be on console.
- Please strictly meet the output format requirements as described in problem statements, because may be some auto judge will be evaluating your program which will compare your output 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.

SAMPLE 1: Pick up students for ICPC Competition.

Atif is driving from Peshawar to the ICPC competition in Lahore. His car has a seating capacity of 7, excluding the driver. On the way, just as he enters the motorway, his mentor Dr. Hussain calls him at 12:00 noon.

Dr. Hussain says,

"Due to a Dharna (sit-in) in Islamabad, all public transport has been stopped and further entry on to the motorway is banned. Please pick-up other students from different motorway interchanges on your way to Lahore, but do not exit, as you will not be able to enter again. Students are waiting in ICPC T-shirts at the interchanges from Peshawar to Lahore. However, these students belong to a school that follows strict hierarchy. Once you pick up a higher ranked student, you cannot pick up a lower ranked one afterwards. Their T-shirts are numbered, and you can pick only one student from an interchange."

The road is one-way, so you cannot take a U-turn. You are required to write a program to guide Atif to pick up maximum number of students in his car. For example, if the students are waiting at different interchanges wearing T-shirts numbered 4, 10, 5, 6, and 8 (arriving in that order), if he picks up the first two, he will not be able to get the other students at 5, 6, 8 because they are lower than 10. However, if he starts picking up students from 5, he can pick up 3 students. However, in the best case he picks up the first student numbered 4, skips the student with shirt numbered 10, and picks up the rest of the 3 students. Your program should help him pick up as many students as possible.

Input:

The input consists of multiple test cases. The first line in the input file is the number of test cases, N. Each of the following N lines contain the total students S waiting, the maximum possible priority number P, and a list of students waiting with arrival order.

Output:

For each test case, print a single line that says "Case# *i*:", where *i* is the test case number, followed by the maximum students picked and their order. A sample input and output format is given below:

Sample Input	Sample Output

2
5 10 1 6 3 5 8
7 10 1 2 3 4 8 5 9
	Case #1: 4 1 3 5 8
	Case #2: 6 1 2 3 4 5 9

SAMPLE 2: Multiply 2 matrices

This sample problem requires you to multiply 2 matrices and give the sum of the resultant matrix. For example, given the matrices M and N, you are required to multiply these matrices. Let A be the resultant matrix. Then your final answer should be the element-wise sum of A.

Note that the dimensions of M and N must match. The only operation allowed is transpose (i.e., changing the rows into columns and columns into rows). If the dimensions of N do not match with those of M, you can try to take the transpose of N.

Example

	1	2	3	4		1	0	3	4		26	52	48
M=	5	6	7	8	N=	5	6	1	8	A=	58	132	96
	9	10	11	12		0	0	0	12		90	212	144

Answer is 26+52+48+26+58+132+96+58+90+212+144+90 = 858

Input

The first line in the input is a single number representing the number of cases your program must process. Each of the subsequent lines then states x_1 and y_1, the number of rows and columns, respectively, of the matrix, M. These values are then followed by the (row-wise) elements of the matrix M. Similarly, (in the same line) we have x_2 and y_2, the number of rows and columns of N followed by its row-wise entries.

Output

For each test case, print a single line that says "Case# i:", where i is the test case number followed by the sum of the resultant matrix. If the matrices cannot be multiplied (even after taking transpose), write "Not possible".

Sample input	Sample Output

2
3 4 1 2 3 4 5 6 7 8 9 10 11 12 3 4 1 0 3 4 5 6 1 8 0 0 0 12
2 3 1 2 3 4 5 6 1 4 1 2 3 4
	Case #1: 858
	Case #2: Not possible

SAMPLE 3: The number game

In this sample problem, you are required to write a program to input two numbers. If both numbers are prime, you should add two numbers. If only one of the numbers is a prime number, you should output their product. If none of the numbers is a prime, then the output is not possible.

Input

The input consists of multiple test cases. The first line of input is the number of test cases, X. Each of the following X lines contain the two numbers x and y.

Output

For each test case, print a single line that says "Case# *i*", where *i* is the test case number, followed by the answer as follows – if both x and y are prime numbers, output x + y; if only one of the numbers is a prime number, output x × y; and, if none of the numbers is a prime number, output "not possible".

Sample Input	Sample Output

2
2 3
3 4
	Case #1: 5
	Case #2: 12