Problem Set

CHALLENGING THE GEEKS OF BALOCHISTAN

Programming Contest

(Pre-Qualifier)

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!

Your program should take input from console output as “Hello %s!”. Sample input/output format is given below:

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 multiple strings separated by lines.

Output:

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

Problem 2: Pick up students for Programming Competition.

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

Dr. Hussain says,

"Due to a Dharna (sit-in) in Mastung, all public transport has been stopped and further entry on to the highway is banned. Please pick-up other students from different highway interchanges on your way to Quetta, but do not exit as you will not be able to enter again. Students are waiting in CGB T-shirts at the interchanges from Khuzdar to Quetta. 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. 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. In case there are multiple possibilities, select the case with the higher values.

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 8 9

Problem 3: 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

M=

1	2	3	4
5	6	7	8
9	10	11	12

N=

1	0	3	4
5	6	1	8
0	0	0	12

A=

26	52	48
58	132	96
90	212	144

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

Problem 4: The number game

In this sample problem, you are required to write a program to input two numbers – x and y. If both numbers are prime, you should add the two numbers. If only one of the numbers is prime, 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, N. Each of the following N 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 $x \times y$; 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

Problem 5: Denominations

Time limit: 5 seconds

Faisal works as a cashier/teller at a local bank located in a not-so-well-off neighborhood. Customers come along all day long, making a good turn over. Every once in a while, an old lady of the neighborhood would cash a cheque. She requests him to give her the minimum number of currency notes, down to 10-rupee bills. For the rest, he gives her toffees that the old lady would give to her children.

Input

The first line of the input contains *n*, which is the number of test cases ($1 \leq n \leq 1000$). Each subsequent line contains the amount to be given to the old lady (in the range $10 - 1{,}000{,}000$).

Output

Each line consists of the test case given by "Case #i: ", where *i* is the test case number. For each test case, output the minimum count of each denomination starting from the largest non-zero denomination. It should then list all the remaining denominations, even if zero, as shown in the sample output. The possible denominations are 5000, 1000, 500, 200, 100, 50, 20, and 10, respectively.

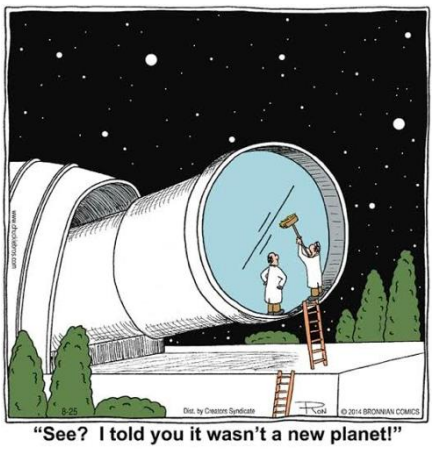
Sample input	Sample Output
2 4270 320	Case #1: 4x1000, 0x500, 1x200, 0x100, 1x50, 1x20, 0x10 Case #2: 1x200, 1x100, 0x50, 1x20, 0x10

Problem 6: Dusty Planet

Time limit: 5 seconds

Capsule Corporation has invented a new telescope to start their observation of the sky. As the world is already familiar with the Saiyan race now and how they reached Earth, that had some pros and cons over time. Now people of Earth want to monitor more closely for any spaceships coming towards their planet.

A kid at the Corporation asked Dr. Brief, the head of Corp, if he can work at night and observe any upcoming spaceships. But during the work, he got distracted and fell in love with astronomy by pointing the telescope at Orion nebula and other breathtaking cosmic objects. He was also getting bored by waiting for spaceships for many days, that might take years by the way. So, he thought he may find a new planet earlier. So, one night when he was looking for undiscovered planets and was very sleepy, he confused a dust particle on the lens of the telescope with a planet. Poor earthling!


"See? I told you it wasn't a new planet!"

To his sleepy excitement, he called Dr. Brief who cleared his misunderstanding. Go home kid, you need some rest today! Tomorrow, I will tell you a trick. So, the next day Dr. Brief pointed a camera at the lens to track the coordinates of any dust particles. You have to help this kid to write a program to find planets but also be careful of the dust.

Input

First Line is a number, *T*, denoting the number of test cases. In each test case, you are given *m* and *n* – the rows and columns of the matrix. The next *m* lines contain an *m***n* 2-D matrix (i.e., each *m* line has *n* entries) showing a capture of space from the telescope. A value of 0 representing space, a value of 1 representing parts of a galaxy, and 2 representing new planets inside galaxies. Galaxies can have size larger than 1×1 if a mesh of consecutive elements are connected in a left-right or top-down fashion only. The next *m* lines contain an *m***n* 2-D matrix showing dust particles as 1.

Output

Output all the galaxies with their starting points and number of new planets in them, each on a separate line. Exclude the planets that are dust particles indeed.

Sample input	Sample Output
1 4 4 2 1 0 0 1 2 0 0 0 0 1 0 1 2 0 2 0 0 1 1 0 0 0 1 0 1 0 1 0 0 0 0	Galaxy starting at 0 0 has 2 planets. Galaxy starting at 2 2 has 0 planets. Galaxy starting at 3 0 has 1 planet. Galaxy starting at 3 3 has 1 planet.

PROBLEM 7: Anagram

Time limit: 3 seconds

An anagram is a word or phrase formed by rearranging the letters of a different word or phrase, typically using all the original letters exactly once. Keeping the concept of anagram in mind this problem is designed.

In this problem, you are given two different strings. Your task is to determine if the second string can be formed using only the letters from the first string. Each letter in the first string can only be used once. There will be no punctuation in either string. For example, given the string "Flowers are blooming" can the sentence "loom ring lower" be formed? The answer is yes. However, given the list "Flowers are blooming" you can't form the sentence "bugs are problems".

Input

The first line gives the number of test cases. Each case has two strings, given on separate lines, with each string having at least one character. The second string may be longer, shorter, or the same length as the first string. All letters input will be lowercase. Read all the data from the console.

Output

Display the word "possible" if the second string can be created using the letters contained in the first string. Display the word "not possible" if the second string can't be created using the letters contained in the first string. Display the output on the monitor.

Sample input & output

This example has 2 test cases. The first test case has the string "apple is a healthy fruit" as the first string and "heal tart has" as the second string, and its answer is "Possible".

Sample Input	Sample Output
2 apple is a healthy fruit heal tart has it is cold driving car	Possible Not Possible