Code Sample Problem

The first stop for Tim the time-traveling salesman is the future. Upon arrival, Tim encounters many different sentient aliens, each having a different number of fingers than the last. He also notices a lot of weird currencies changing hands.

It turns out that counterfeiting is a serious problem in the future, so much so that the intergalactic government has gone overboard in minting new and different types of currency. Since the members of government are diverse in the number of fingers that they have, and everyone has very good math skills, the new currencies have extremely confusing denominations. Rather than a Melka being worth 10 Zedars and a Zedar being worth 10 Milbacs, a Melka is worth 13 Zedars and a Zedar is worth 41 Milbacs (the alien race known as the Stalsgeap are famous for their 41 fingered hands).

Though Tim is going to have trouble keeping all of these currencies straight, one thing he knows for sure is to buy low and sell high. Given the currency denominations and different vendors' prices for an item in terms of the number of currency notes required to buy it, Tim needs to figure out the difference between the least price he can buy the item for and the most he can sell it for.

Please provide an application showing use of the MVC design pattern with supporting unit tests. Also note that there may or may not be test cases outside of the sample input provided that should be handled if possible.

Input

The first line is the number K of input data sets, followed by the K data sets, each of the following form:

The first line of each data set contains the number 2 <= D <= 7 of denominations, and the number 2 <= N <= 10 of the item's different prices. On the next line are D −1 positive integers, where the i^{th} integer specifies the number of notes of denomination i + 1 that are equivalent to 1 note of denomination *i*.

This is followed by N lines, each one specifying a price in terms of the quantity of notes of each denomination. Each line contains D non-negative integers, where the ith integer specifies the number of notes of denomination *i*. It is guaranteed that each of the N prices, in terms of the smallest denomination, will fit in a 32-bit integer.

Output

For each data set, output "`Data Set x:`" on a line by itself, where `x` is its number. On the next line, output the difference between the highest price and the lowest price, in terms of the smallest denomination. Each data set should be followed by a blank line.

Sample Input / Output

Input	Output
2 2 2 2 2 0 0 5 3 3 3 5 1 1 1 3 0 0 1 10 0	Data Set 1: 1 Data Set 2: 44