Chocolate

We are given a bar of chocolate composed of $m*n$ square pieces. One should break the chocolate into single squares. Parts of the chocolate may be broken along the vertical and horizontal lines as indicated by the broken lines in the picture.

A single break of a part of the chocolate along a chosen vertical or horizontal line divides that part into two smaller ones. Each break of a part of the chocolate is charged a cost expressed by a positive integer. This cost does not depend on the size of the part that is being broken but only depends on the line the break goes along. Let us denote the costs of breaking along consecutive vertical lines with $x_1, x_2, ..., x_{m-1}$ and along horizontal lines with $y_1, y_2, ..., y_{n-1}$.

The cost of breaking the whole bar into single squares is the sum of the successive breaks. One should compute the minimal cost of breaking the whole chocolate into single squares.
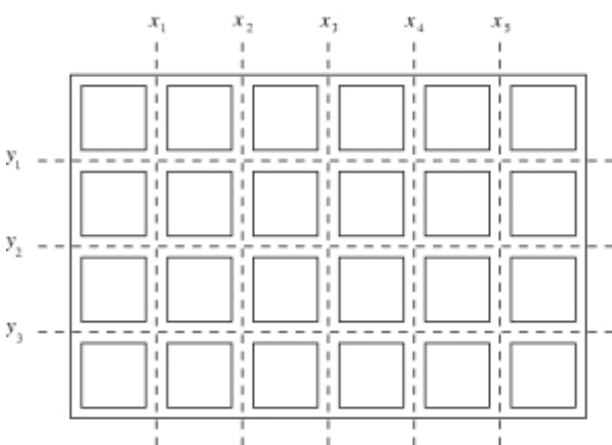
For example, if we break the chocolate presented in the picture first along the horizontal lines, and next each obtained part along vertical lines then the cost of that breaking will be $y_1+y_2+y_3+4*(x_1+x_2+x_3+x_4+x_5)$.

Task

Write a program that for each test case:

- Reads the numbers $x_1, x_2, ..., x_{m-1}$ and $y_1, y_2, ..., y_{n-1}$
- Computes the minimal cost of breaking the whole chocolate into single squares, writes the result.

Input

One integer in the first line, stating the number of test cases, followed by a blank line. There will be not more than 20 tests.

For each test case, at the first line there are two positive integers m and n separated by a single space, $2 <= m,n <= 1000$. In the successive m-1 lines there are numbers $x_1, x_2, ..., x_{m-1}$, one per line, $1 <= x_i <= 1000$. In the successive n-1 lines there are numbers $y_1, y_2, ..., y_{n-1}$, one per line, $1 <= y_i <= 1000$.

The test cases will be separated by a single blank line.

Output

For each test case : write one integer - the minimal cost of breaking the whole chocolate into single squares.

Example

Input:
1

6 4
2
1
3
1
4
4
1
2

Output:
42