## Exercise – *DHL*

You have recently been employed by the IT department of DHL. Your first assignment is to help them optimise the parcel collection process in the warehouse. To simplify, we model the problem as follows. There are two stacks of $n$ parcels each, one containing very bulky items (stack $A$) and the other very heavy items (stack $B$). The volumes of the parcels on stack $A$ are denoted by integers $a_1, \ldots, a_n$, where $a_i$ is the volume of i-th parcel from the bottom. For example, $a_1$ is the volume of the parcel on the bottom of $A$, and $a_n$ of the parcel on the top. Similarly, the weights of the parcels on stack $B$ are denoted by integers $b_1, \ldots, b_n$, again from bottom to top.

A machine that manipulates parcels can only access the parcel of the top of each stack. Each time a truck comes to collect parcels, the machine loads a certain number of parcels from both $A$ and $B$. For example, the machine can load the top two parcels from $A$ (pick up the top one, load it, pick up the next one, load it) and the top three parcels from $B$. After that the truck leaves and a new one comes. This process is repeated until there are no more parcels left. Due to a malfunction in the software, the machine must load at least one parcel from $A$ and one from $B$ each time, otherwise it breaks and you get fired! In particular, both stacks must become empty in the same round.

The cost of one loading is equal to $(S_a - k_a) \cdot (S_b - k_b)$, where $S_a$ is the sum of volumes of parcels loaded from $A$, $S_b$ is the sum of weights of parcels loaded from $B$, and $k_a$ and $k_b$ are the number of parcels loaded from $A$ and $B$, respectively. This formula represents the difficulty of handling bulky and heavy things at the same time. For example, if in the first round the machine loads the top two parcels from $A$ and the top three parcels from $B$, the cost of such loading is $(a_n + a_{n-1} - 2) \cdot (b_n + b_{n-1} + b_{n-2} - 3)$. Then, if in the second round it loads one parcel from $A$ and two from $B$, the new cost is $(a_{n-2} - 1) \cdot (b_{n-3} + b_{n-4} - 2)$, since these parcels are now on top of the stack. Your job is to schedule loading as such that the sum of all costs is the smallest possible without getting fired.

**Input** The first line of the input contains the number of test cases $t \leqslant 30$. Each of the $t$ test cases is described as follows.

- It starts with a line that contains one integer $n$, denoting the number of parcels on each of the tracks ($1 \leqslant n \leqslant 1000$).
- The following line contains $n$ integers $a_1 \ \ldots \ a_n$, separated by a space, where $a_i$ denotes the volume of i-th parcel in $A$ from the bottom ($1 \leqslant a_i \leqslant 100$).
- The following line contains $n$ integers $b_1 \ \ldots \ b_n$, separated by a space, where $b_i$ denotes the weight of i-th parcel in $B$ from the bottom ($1 \leqslant b_i \leqslant 100$).

**Output** For every test case the corresponding output appears on a separate line. It consists of one integer, the smallest possible sum of costs required to load all parcels without getting fired.

**Points** There are four groups of test sets, worth 100 points in total.

1. For the first group of test sets, worth 20 points, you may assume that $n \leqslant 6$.
2. For the second group of test sets, worth 40 points, you may assume that $n \leqslant 100$.
3. For the third group of test sets, worth 39 points, you may assume that $n \leqslant 300$.
4. For the forth group of test sets, worth 1 point, there are no additional assumptions. This is a challenging test set, and you should only try it for your own enjoyment!

**Sample Input**

```
3
5
1 2 3 4 5
5 4 3 2 1
8
1 2 1 3 1 4 1 5
4 3 2 1 8 7 6 5
8
4 6 6 8 7 5 4 8
10 13 3 5 4 6 8 7
```

**Sample Output**

```
0
16
211
```

this is equivalent to sumation of area. You can easily show that, for optimal solution, each time one of the parcel is 1.

another thing is that, whether a or b have width 1, this cannot happen continuously, it must switch, or 11.

**Explanation** One way to achieve the optimal solution in the third test case is to first remove one from each, then one from A and two from B, then one from each, and again one from each, then three from A and one from B, and finally one from A and two from B.