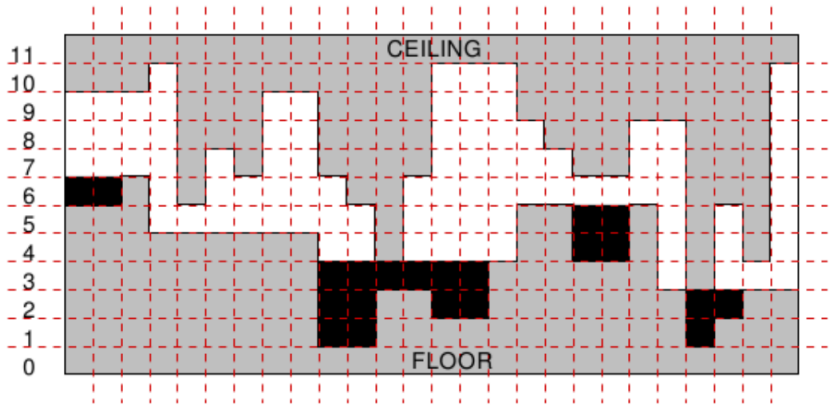
P5. Cave

(Time Limit: 3 seconds)

As an owner of a land with a cave you were delighted when you last heard that underground fuel tanks are great business. Of course, the more volume one can store, the better. In case of your cave, the efflective volume is not easy to calculate, because the cave has a rather sophisticated shape (see figure). Thank heavens it is degenerate in one dimension!



The cave. All ponds that can be flooded with fuel are marked black.

Furthermore, there is some electrical wiring on the ceiling of the cave. You can never be sure if the insulation is intact, so you want to keep the fuel level just below the ceiling at every point. You can pump the fuel to whatever spots in the cave you choose, possibly creating several ponds. Bear in mind though that the fuel is a liquid, so it minimises its gravitational energy, e.g., it will run evenly in every direction on a flat horizontal surface, pour down whenever possible, obey the rule of communicating vessels, etc. As the cave is degenerate and you can make the space between the fuel level and the ceiling arbitrarily small, you actually want to calculate the maximum possible area of ponds that satisfy aforementioned rules.

Input

The input contains several test cases. The first line of the input contains a positive integer $Z \leq 15$, denoting the number of test cases. Then Z test cases follow, each conforming to the format described below.

In the first line of an input instance, there is an integer n ($1 \leq n \leq 10^6$) denoting the width of the cave. The second line of input consists of n integers $p_1, p_2, \ldots, p_n$ and the third line consists of n integers $s_1, s_2, \ldots, s_n$, separated by single spaces. The numbers pi and s_i satisfy $0 \leq pi < s_i \leq 1000$ and denote the floor and ceiling level at interval [i, i + 1), respectively.

Output

For each test case, your program is to print out one integer: the maximum total area of admissible ponds in the cave.

Sample Input

1

15

6 6 7 5 5 5 5 5 5 1 1 3 3 2 2

10 10 10 11 6 8 7 10 10 7 6 4 7 11 11

Sample Input

14

P6. The Sultan's Successors

(Time Limit: 3 seconds)

The Sultan of Nubia has no children, so she has decided that the country will be split into up to k separate parts on her death and each part will be inherited by whoever performs best at some test. It is possible for any individual to inherit more than one or indeed all of the portions. To ensure that only highly intelligent people eventually become her successors, the Sultan has devised an ingenious test. In a large hall lled with the splash of fountains and the delicate scent of incense have been placed k chessboards. Each chessboard has numbers in the range 1 to 99 written on each square and is supplied with 8 jewelled chess queens. The task facing each potential successor is to place the 8 queens on the chess board in such a way that no queen threatens another one, and so that the numbers on the squares thus selected sum to a number at least as high as one already chosen by the Sultan. (For those unfamiliar with the rules of chess, this implies that each row and column of the board contains exactly one queen, and each diagonal contains no more than one.)

Write a program that will read in the number and details of the chessboards and determine the highest scores possible for each board under these conditions. (You know that the Sultan is both a good chess player and a good mathematician and you suspect that her score is the best attainable.)

Input

Input will consist of ***k*** (the number of boards), on a line by itself, followed by ***k*** sets of 64 numbers, each set consisting of eight lines of eight numbers. Each number will be a positive integer less than 100. There will never be more than 20 boards

Output

Output will consist of ***k*** numbers consisting of your ***k*** scores, each score on a line by itself and right justied in a eld 5 characters wide.

Sample Input

```
1
1 2 3 4 5 6 7 8
9 10 11 12 13 14 15 16
17 18 19 20 21 22 23 24
25 26 27 28 29 30 31 32
33 34 35 36 37 38 39 40
41 42 43 44 45 46 47 48
48 50 51 52 53 54 55 56
```

57 58 59 60 61 62 63 64

Sample Output

260