The House Always Wins

Program Name: `House.java` **Input File: `house.dat`**

You just started work for Casinos 'R Us, and have been given your first task. The Marketing department has created a new game, but the Finance department is freaking out over potential losses. You are to write a program to calculate the maximum potential winnings; your program will be a component of the larger game program. This electronic game, *Pachinko Path*©, will randomly assign dollar amounts to pins on the board. When the game ball falls, each pin touched will be added to the total value of that play. The rules get complicated after that, but the concern is that too high a value will cause a loss to the casino. After your program reports the maximum value, new random values may be chosen until a "better" board is set.

```
        7
      3   8
    8   1   0
  2   7   4   4
4   5   2   6   5
```

(Figure 1)

Figure 1 shows a possible pattern (always a triangle) of pins with dollar values. Write a program that calculates the highest sum of numbers passed on a route that starts at the top and ends somewhere on the base. Each step can go either diagonally down to the left or diagonally down to the right. The highest dollar value on the bottom row is what your program will report.

Input

The input file consists of a single integer *c*, which is the number of test cases, $1 \leq c \leq 10$. Following that is *c* sets of data, where each data set is a sequence of lines of integers. The first line in each set contains one integer *n*: the number of rows in the triangle. The following *n* lines describe the data of the triangle. The number of rows in the triangle is > 1 but ≤ 100. The numbers in the triangle, all integers, are between 0 and 99.

Output

Each output line contains a single integer, which is the maximum drinkable volume possible for that input.

Your program is to write to standard output, one line for each input test case. The highest possible dollar value is written as an integer.

Sample Input

```
1
5
7
3 8
8 1 0
2 7 4 4
4 5 2 6 5
```

Sample Output

```
30
```