Tricky Tolls

Filename: `tolls`

The Problem

Stephen loves traveling, but hates toll roads. Well, not so much the roads, but the tolls[1]. He happens to live next to a very wacky toll road. In an effort to confuse commuters, the state has imposed a variable fee at each toll booth. The fee Stephen pays at a toll both depends upon what time he goes through. If he goes through in the first half of an hour (5:00 to 5:29, for example), then he pays the *early* toll. If he goes through in the second half of an hour (5:30 to 5:59, for example), then he pays the *late* toll.

However, Stephen is not a naïve traveler. He knows that it is possible for him to save money by planning his trip around these crazy tolls. For instance, let's say it's 7:25 and Stephen is at a toll booth. The early toll is $10, and the late toll is $7. In this case it's less money for Stephen to just stop at this toll booth, wait 5 minutes until is it 7:30, and then pay the late toll of $7 instead of the early toll of $10. But if Stephen always does this, he could end up making some *very* long trips. For this reason, Stephen adopts the "time is money" attitude. Stephen values his time greatly, so he gives a value of $1 to every minute spent waiting at a toll booth. This means that from the example above, the total cost of waiting for the late toll will actually be $7 + $5 = $12, so it would be better to just go ahead and pay $10 at 7:25 for the early toll and continue driving.

Now Stephen wants you to write a program to figure out the 'cheapest' schedule he can make to get from the first toll booth through the last toll booth, where the cost is the sum of tolls plus total time spent waiting. Stephen cannot skip any toll booth, and must travel them in order. He has a very open schedule, and can arrive at the first booth at any time.

The Input

There will be several test cases. Input will begin with a single positive integer T giving the number of test cases. Each test case will begin with a positive integer N ($2 \leq N \leq 100$) indicating the number of toll booths. The next line will contain exactly N-1 integers, giving the amount of time in minutes it takes to travel between the toll booths. The first number will be the time to go from the 1st to the 2nd booth, the second number will be the time to go from the 2nd to the 3rd booth, etc., and the (N-1)th number will be the time to go from the next to last toll booth to the last toll booth. The last N lines of each input case will have 2 integers each, giving the *early* and *late* toll costs, respectively, for that particular toll booth. All times and costs will be non-negative integers ≤ 1000.

The Output

For each case, output a single line

```
Case #X: Y
```

where X is the case number beginning with 1, and Y is the 'cheapest' cost possible.

---

[1] Largely due to the fact that he is a poor college student.

Sample Input

```
3
2
25
10 20
15 30
2
25
10 20
50 10
2
60
10 20
40 15
```

Sample Output

```
Case #1: 25
Case #2: 20
Case #3: 26
```