Given a set $S = (e_1, e_2, ..., e_n)$ of n distinct elements such that $e_1 < e_2 < \ldots < e_n$ and considering a binary search tree (see the previous problem) of the elements of S, it is desired that higher the query frequency of an element, closer will it be to the root.

The cost of accessing an element e_i of S in a tree ($cost(e_i)$) is equal to the number of edges in the path that connects the root with the node that contains the element. Given the query frequencies of the elements of S, $(f(e_1), f(e_2), \ldots, f(e_n))$, we say that the total cost of a tree is the following summation:

$$f(e_1) * cost(e_1) + f(e_2) * cost(e_2) + \ldots + f(e_n) * cost(e_n)$$

In this manner, the tree with the lowest total cost is the one with the best representation for searching elements of S. Because of this, it is called the Optimal Binary Search Tree.

Input

The input will contain several instances, one per line.

Each line will start with a number $1 \leq n \leq 250$, indicating the size of S. Following n, in the same line, there will be n non-negative integers representing the query frequencies of the elements of S: $f(e_1), f(e_2), \ldots, f(e_n)$, $0 \leq f(e_i) \leq 100$. Input is terminated by end of file.

Output

For each instance of the input, you must print a line in the output with the total cost of the Optimal Binary Search Tree.

Sample Input

```
1 5
3 10 10 10
3 5 10 20
```

Sample Output

```
0
20
20
```