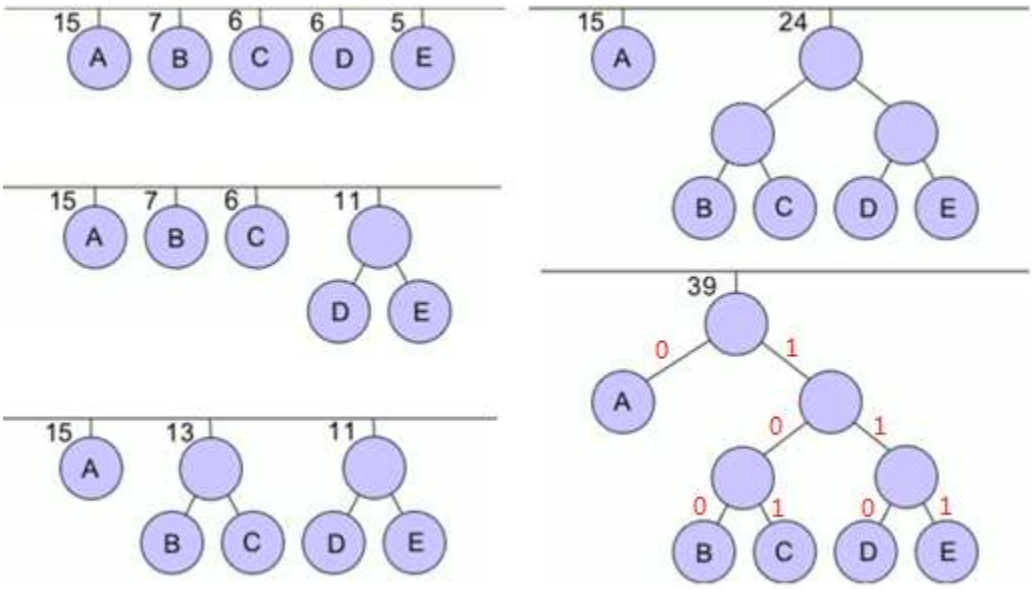
Huffman Coding

Huffman code is a particular type of optimal prefix code that is commonly used for lossless data compression. The idea of Huffman coding is quite simple: encode frequently used symbols with fewer bits. Huffman code is produced from the Huffman tree, which is a binary tree.

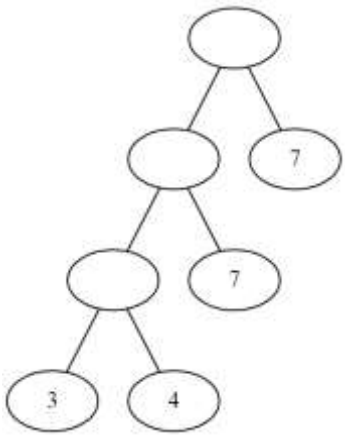
Initially, we extract the frequency (occurrence count) of each symbol from the text file. In each step, we merge two sub-trees of the least frequency into a new binary tree, until the final binary tree is constructed. For each symbol, its Huffman code is the path from the root to the symbol node where we encode left-tree as '0' and right-tree as '1'. Here is an example of constructing Huffman tree:



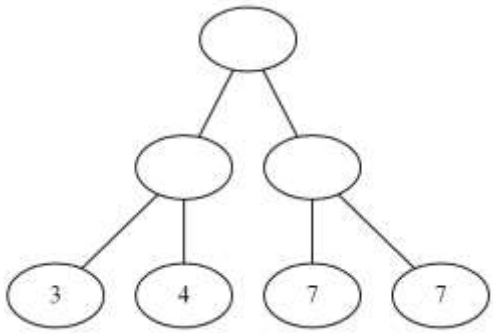
Symbol	A	B	C	D	E
Frequency	15	7	6	6	5
Code	0	100	101	110	111

In this question, given the frequency of each symbol, can you calculate the total weighted length of the Huffman code of all symbols? For example the weighted length of the above Huffman tree is: 15×1+7×3+6×3+6×3+5×3=87

Well, the Huffman tree might not be unique when more than two symbols have the same frequency. However, the weighted length is always the same. In the example of four symbols with frequencies {3, 4, 7, 7}, two possible Huffman trees could be constructed as follows:



Weighted length = 3×3+4×3+7×2+7×1 = 42



Weighted length = 3×2+4×2+7×2+7×2 = 42

Input

The input contains multiple test cases. Each test case begins with one integer n ($0 \leq n \leq 100000$), indicating the number of symbols. The following line gives n positive integers $f_1, f_2, \ldots, f_n$ representing the frequency of each symbol where $1 \leq f_i \leq 1000$.

Output

For each test case, print the total weighted length of the Huffman codes in a separate line.

Sample input	Sample output
5 15 7 6 6 5 4 3 4 7 7 6 8 6 5 5 4 3 10 8 1 7 3 6 9 4 2 3 3	87 42 79 144